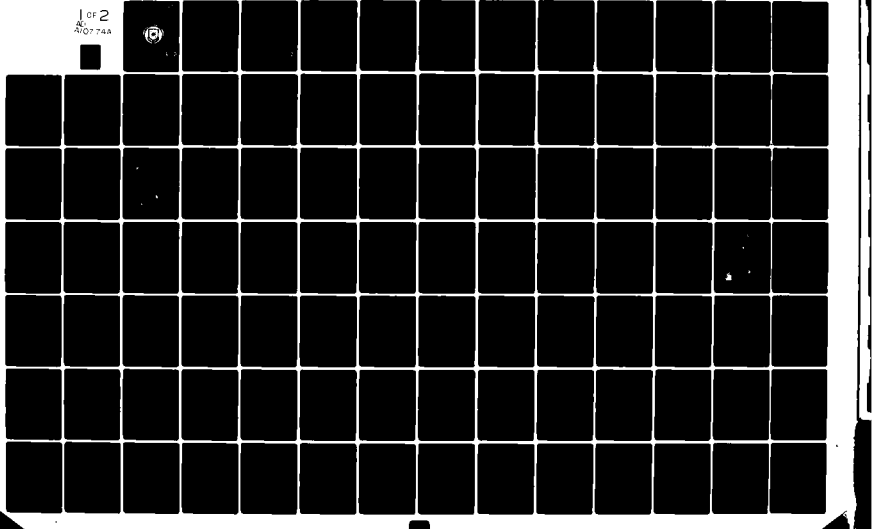
AD-A107 744 LOGICON INC LEXINGTON MA F/G 9/2
REQUIREMENTS ENGINEERING GUIDEBOOK. REQUIREMENTS ENGINEERING US--ETC(U)
JUL 80 D G SMITH DAAG29-76-D-0100
UNCLASSIFIED RSD-RD022-VOL-II AIRMICS-80-8-2 NL



AD A107744

15

AIRMICS

US ARMY INSTITUTE FOR RESEARCH IN MANAGEMENT INFORMATION AND COMPUTER SCIENCE

LEVEL

DTIC FILE COPY

REQUIREMENTS ENGINEERING GUIDEBOOK
REQUIREMENTS ENGINEERING
USING AN AUTOMATED TOOL: PSL/PSA

FINAL REPORT - Volume II AIRMICS 80-8-1

DTIC ELECTE S NOV 25 1981 D D

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Requirements Engineering Guidebook Requirements Engineering Using an Automated Tool: PSL/PSA Volume II		5. TYPE OF REPORT & PERIOD COVERED FINAL 6. PERFORMING ORG. REPORT NUMBER AIRMICS 80-8-2
7. AUTHOR(s) Daniel G. Smith		8. CONTRACT OR GRANT NUMBER(s) DAAG 29-76-D-0100 Delivery Order 1108 TCN 79-018
9. PERFORMING ORGANIZATION NAME AND ADDRESS LOGICON 18 Hartwell Avenue Lexington, MA 02173		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE(62725A) PROJECT NUMBER DY10
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Institute for Research in Management Information and Computer Science O'Keefe Building, GIT, Atlanta, Ga. 30332		12. REPORT DATE July 80 13. NUMBER OF PAGES 100
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Approved for public release; distribution unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

SYSTEM, DATA PROCESSING, REQUIREMENTS, METHODOLOGY PSL/PSA, REQUIREMENTS ENGINEERING

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Based on a general awareness of the inadequacies in requirements definition throughout the life cycle of a system, AIRMICS has initiated a comprehensive program for application of the latest methodologies in requirements analysis to the Army's information system development programs. This report was prepared to aid requirements analysts who are new to automated requirements analysis in using PSL/PSA by selecting a subset of PSL for initial use and in showing the relationship between the manual method and the automated language support.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REQUIREMENTS ENGINEERING GUIDEBOOK

REQUIREMENTS ENGINEERING USING AN AUTOMATED TOOL: PSL/PSA

Final Report - Volume II Logicon Report No. ESD-RC022

July 1980

For The

U.S. ARMY COMPUTER SYSTEMS COMMAND

Accession For	
NTIS GRA&I	X
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

Prepared by

DTIC
ELECTE
S NOV 25 1981 D
D

18 Hartwell Avenue
Lexington, MA 02173

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DISTRIBUTION STATEMENT

Approved for public release. Distribution unlimited

This Technical Report has been reviewed and approved.

Daniel E. Hocking
Computer Scientist, Computer Science Division, U.S. Army Institute for Research in Management Information and Computer Science

John R. Mitchell
Chief, Computer Science Division, U.S. Army Institute for Research in Management Information and Computer Science

Clarence Giese, Director
U.S. Army Institute for Research in Management Information and Computer Science

LOGICON

REQUIREMENTS ENGINEERING GUIDEBOOK

REQUIREMENTS ENGINEERING USING AN AUTOMATED TOOL: PSL/PSA

Final Report - Volume II Logicon Report No. ESD-R0022

July 1980

Author:

Daniel G. Smith

Daniel G. Smith

Reviewed By:

Larry A. Johnson
Martin F. McDonough
Fredrick T. Coker

The views, opinions and recommendations expressed in Volumes I and II of this document are those of the author and do not constitute an official Department of the Army position or policy unless so designated.

PREFACE

In 1973 the Air Force initiated an advanced development project within the Electronics Systems Command (AFSC/ESD) to acquire and apply the ISDOS Project (University of Michigan) Problem Statement Language/ Problem Statement Analyzer (PSL/PSA) to ESD requirements analysis needs. The Air Force version of PSL/PSA was accepted by ESD in 1974 and was called the User Requirements Language/User Requirements Analyzer (URL/URA)*. In 1975 Logicon performed an evaluation for the ESD Joint Surveillance System (JSS) program office on the applicability of URL/URA as a tool for both the analysis of JSS requirements and the design of the JSS. As a result of the JSS study, Logicon began to use URL/URA in its system engineering support role to JSS and is continuing the use of the tool in the development phase of the JSS at this time.

During the past five years Logicon has broadened its experience using URL/URA. Additional studies, applications, and training programs for industry and government have been performed. A formalized approach has been developed by Logicon for applying URL/URA and Logicon has extended URL/URA under contract to the Air Force. In addition, Logicon has independently translated and installed URL/URA and the Logicon extensions and modifications on the Logicon VAX 11/780 system in Lexington, MA. Additional extensions for such applications as automated specification documentation generation from the URL/URA data bases have been developed and applied to new projects within the past year.

This guidebook was developed under contract to the U.S. Army Computer Systems Command (USACSC), Army Institute for Research in Management Information and Computer Science (AIRMICS). It is the second of a two volume report for USACSC and has been prepared to aid the systems requirements definition and analysis process (requirements engineering) within USACSC. Guidelines and standards for requirements engineering and the use of PSL/PSA are presented in this guidebook.

---

* URL/URA was also previously called the Computer-Aided Requirements Analyzer (CARA) by the Air Force. URL/URA is currently referred to as the Computer-Aided Design,Specification, and Analysis Tool (CADSAT) by the Air Force.

TABLE OF CONTENTS

		Page
PREFACE		ii
LIST OF FIGURES		
LIST OF TABLES		
1.	INTRODUCTION	1
1.1	Purpose	1
1.2	Scope	1
1.3	Definitions	1
1.3.1	System	1
1.3.2	Requirements Engineering	1
1.3.3	Quality Requirements	2
1.4	Contents	2
2.	QUALITY REQUIREMENTS CHARACTERISTICS	4
2.1	Introduction	4
2.2	Discrete Requirements	4
2.3	Organization of Requirements	6
2.3.1	Hierarchical Organizational Relationships	6
2.3.2	System Flow Relationships	9
2.3.3	Requirements Traceability Relationships	12
2.4	Summary	13
3.	STRUCTURED APPROACHES AND AUTOMATED TOOLS	14
3.1	Structured Approaches	14
3.2	ISDOS Project -PSL/PSA	14
3.3	Utility of PSL/PSA	15
3.4	Guidelines for Requirements Engineering Using PSL/PSA	16
4.	REQUIREMENTS ENGINEERING PROCEDURES	17
4.1	Introduction	17
4.2	Identify and Review Source Documentation BLOCK 1	23
4.3	Produce Requirements Engineering Plan BLOCK 2	23
4.4	Identify System Functions BLOCK 3	24
4.5	Organize Functions into a Hierarchical Structure BLOCK 4	25
4.6	Identify System Constraints BLOCK 5	26
4.7	Identify System Using Activities BLOCK 6	29

4.8	Identify External System Inputs-Outputs	BLOCK 7 . .	30
4.9	Perform Information-Flow Analysis	BLOCK 8 . .	31
4.10	Structure System Information	BLOCK 9 . .	34
4.11	Perform Control-Flow Analysis	BLOCK 10 .	36
4.12	Perform Test Analysis	BLOCK 11 .	39
4.13	Prepare Specification Documentation	BLOCK 12 .	40
4.14	Perform Traceability Analysis	BLOCK 13 .	42
4.15	Perform Consistency and Completeness Analysis	BLOCK 14 .	46
4.16	Manage Requirements Engineering Activities	BLOCK 15 .	51

APPENDICES

Appendix A - Selected References 53
Appendix B - Selected Language Features 55
Appendix C - Abstracts of Analyzer Reports 72
Appendix D - Example Analyzer Reports 82

LIST OF FIGURES

Page

1. Development of Discrete and Well-Organized Requirements 5
2. Functional Hierarchical Structure 7
3. I/O Hierarchical Structure 8
4. Control-Flow Diagram 10
5. Information-Flow Diagram 11
6. Requirements Definition and Analysis 20
7. Requirements Engineering Procedures 21
8. Information-Flow Diagram with PSL constructs 33
9. Control-Flow Diagram with PSL constructs 37
10. Requirements Traceability Analysis 43
11. Requirements Configuration Control 44

LIST OF TABLES

1. System Requirement Types 18

SECTION 1 INTRODUCTION

1.1 Purpose

This requirements engineering guidebook provides guidance and standards for defining and analyzing the requirements for a system using an automated tool (PSL/PSA). This guidebook addresses the modeling of the functional requirements of a system (logical modeling) during the initial phases of a system acquisition, the definition phase. The guidance can be applied to large-scale as well as smaller, less complex systems and can be used in various acquisition environments.

1.2 Scope

This guidebook is compatable with modern structured approaches to requirements definition and analysis and provides guidance on the selected use of PSL/PSA. References to documentation on various modern structured approaches and to PSL/PSA are provided. The user of this guidebook may follow the approach presented herein or tailor the approach to emphasize a particular feature of any of the modern structured analysis methods cited in Appendix A.

1.3 Definitions

1.3.1 System

A composite of items, assemblies, skills, and techniques capable of performing and/or supporting a using organizations' needs. A complete system includes related facilities, items, material, services, and personnel required for its operation to the degree that it can be considered a self-sufficient item in its intended use.

1.3.2 Requirements Engineering

Requirements Engineering is an iterative process of defining the system requirements and analyzing the integrity of the requirements. This process involves all areas of system development preceding the actual design of the system. The products of the requirements engineering process can be evaluated for completeness, consistency, testability, and traceability. The essential goal of requirements engineering is to thoroughly evaluate the needs which the system must satisfy.

1.3.3 Quality Requirements

The term 'quality requirements' is used throughout this guidebook to denote system requirements which are complete, consistent, testable, and traceable. This characteristic is the result of the requirements being discretely identified and well-organized as discussed in the sections to follow.

1.4 Contents

The remainder of this guidebook consists of three sections and four appendices, as follows:

Section 2 - Quality Requirements Characteristics: Provides a description of the two requirements characteristics: discrete and well organized. This discussion is followed by a description of three forms of well-organized requirements: hierarchical structures, system flows, and requirements traceability.

Section 3 - Structured Approaches and Automated Tools: Briefly describes the trend in structured analysis approaches and the use of an automated tool, PSL/PSA. The utility of PSL/PSA as a requirements engineering tool is presented along with the various versions of PSL/PSA.

Section 4 - Requirements Engineering Procedures: Provides the procedural framework for defining and analyzing system requirements. The procedures consist of fifteen activities which are explained in functional terms: namely, activities to be performed by requirements engineers. The language and report features of PSL/PSA which support each activity are presented.

Appendix A - Selected References: Provides a list of references primarily consisting of structured approaches to requirements engineering and references of the pertinent versions of PSL/PSA.

Appendix B - Selected Language Features: Provides a condensed list of PSL features which support the requirements engineering activities presented in Section 4. Cross references are provided to Section 4 and to the various versions of PSL.

Appendix C - Analyzer Reports: Provides a list of PSA reports which support the requirements engineering activities presented in Section 4. Cross references are provided to Section 4 and to the various versions of PSA.

Appendix D - Example Analyzer Reports: Provides examples of some of the PSA reports described in Appendix C.

SECTION 2 QUALITY REQUIREMENTS CHARACTERISTICS

2.1 Introduction

Quality requirements are dependent upon the analyst first identifying the discrete requirements of the system and then organizing these requirements in effective ways for further analysis. The resulting organization is often referred to as a functional or logical model, where the objects in the model and their relationships to each other provide a comprehensive description (specification) of the user needs.

Initial documentation for identifying the user's system requirements may include early planning documents and specifications for similar systems, for system interfaces, and for existing or previously defined subsystems. In addition, documentation derived from engineering studies and prototyping or experimental test systems may be available. If the engineering activities have advanced beyond the planning and study stage, specification documents may have already been developed. These early requirements documents usually have one prevailing characteristic: The system requirements are not typically distinguished (discrete) or collectively defined (well-organized).

2.2 Discrete Requirements

Figure 1 illustrates the first characteristic of quality requirements: discreteness. The key to identifying discrete requirements is to break the user needs into individual parts (objects) which represent non-overlapping requirements. Discrete requirements include system objects (functions, external and internal inputs and outputs, etc.) and properties (statements about the objects) such as constraints and descriptions. At this point missing or incomplete requirements can be more readily identified. This itemization and categorization of requirements introduces clarity, whereas the sources of the requirements may be overstated, ambiguous, redundant, incomplete, and inconsistent. This process of itemization also provides the basis for verifying the quality of the requirements and for assessing the ability to test the requirements in the target system.

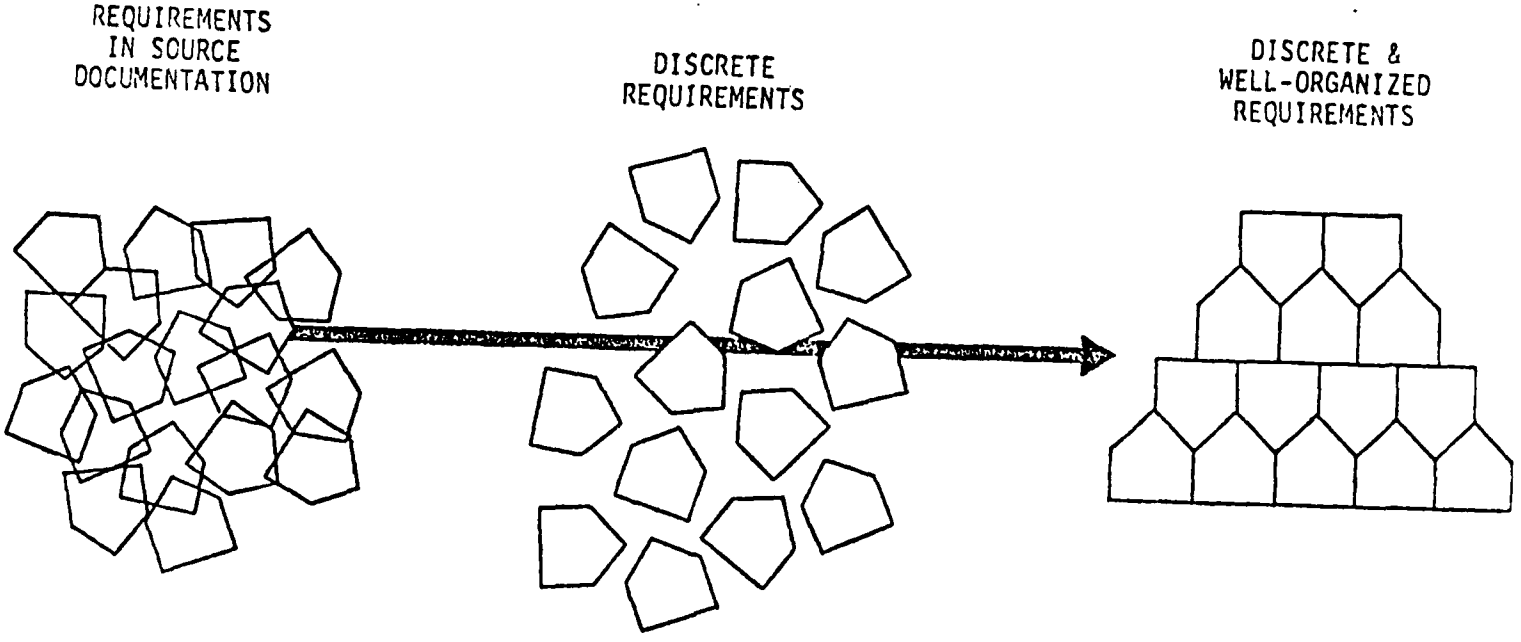


Figure 1. Development of Discrete and Well-Organized Requirements

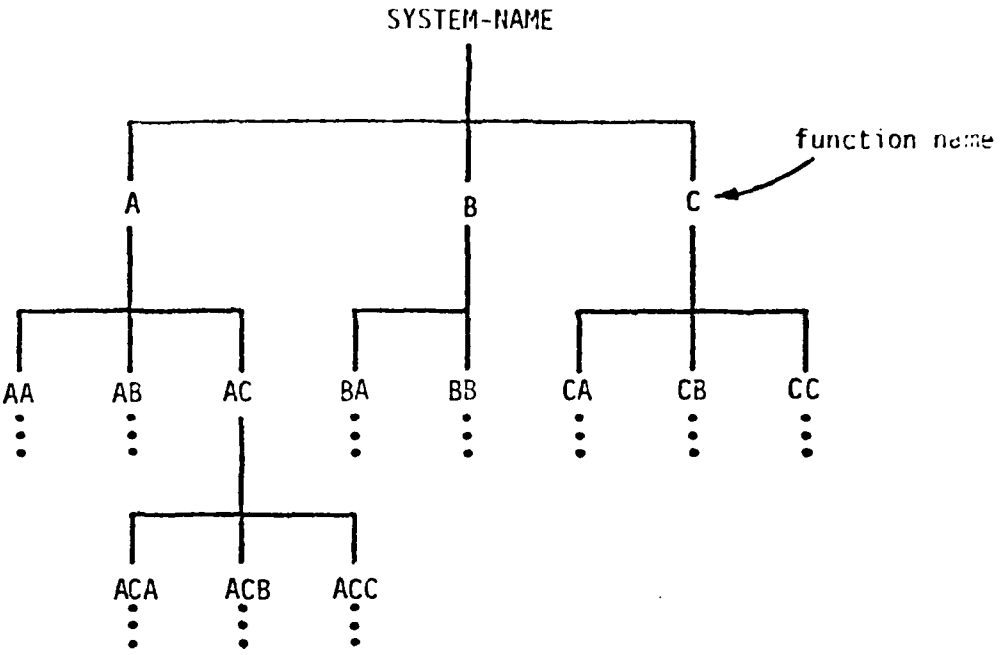
2.3 Organization of Requirements

The second characteristic of a good statement of requirements is the arrangement of the requirements in effective ways for additional analysis and for communicating these requirements to the using agency and to design engineers. The identification of discrete requirements provides some awareness of omissions and gaps in the requirements. This awareness is further heightened by organizing the requirements in ways which identify all the relationships among the discrete requirements (Figure 1). These relationships are of three types: hierarchical organizational relationships, system flow relationships, and requirements traceability relationships. The following paragraphs discuss these relationships.

2.3.1 Hierarchical Organizational Relationships

Hierarchical organizational relationships are shown by structuring the discrete functions and the information requirements (external and internal inputs and outputs) of the system into hierarchical structures. The concept of a functional hierarchical structure (Figure 2) was introduced into military systems development through initial systems engineering practices dating back to the 1940s. This concept has been maintained in military systems development and documentation throughout the 1960s and is an integral part of the current military standards for system documentation, i.e., DoD Standard 7935.1-S [1] and MIL-STD-490 [2]. This form of organization provides a view of the system as an aggregate of functions broken into a logical arrangement of subordinate discrete activities which must be performed. Over the course of requirements engineering many missing or incomplete functions can be directly identified from the functional hierarchical structure.

The discrete system inputs, outputs (external I/O) and the internal information requirements (internal I/O) necessary for the system's operation can be similarly structured (data structures). The emphasis again is the arrangement of the information requirements into hierarchical structures by breaking the information into logical subordinate parts or groupings (Figure 3). A well-organized data structure is effective in communicating the information requirements and for identifying incomplete or missing information requirements.

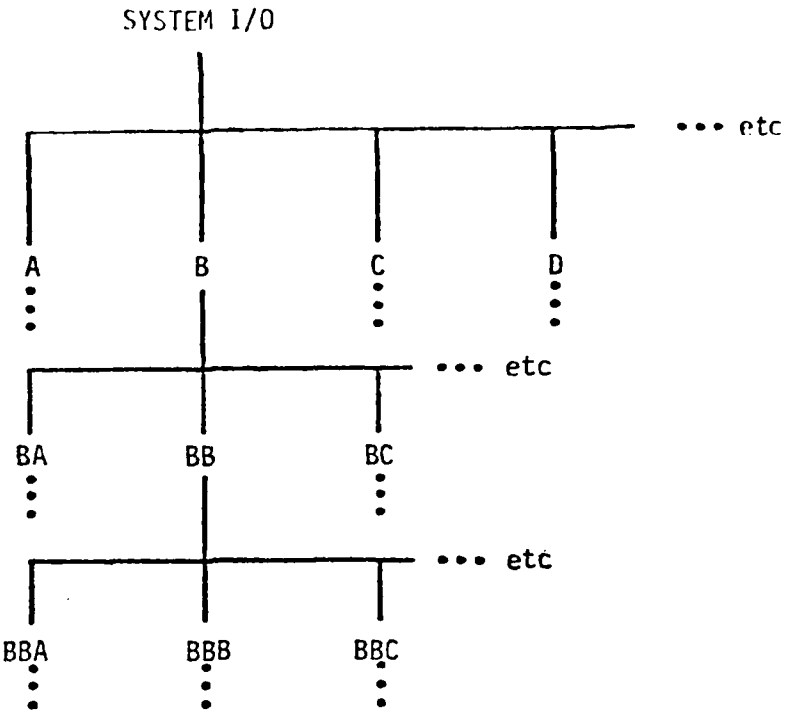


Graphic Representation

```
SYSTEM-NAME
  A
     AA ...
     AB ...
     AC ...
       ACA ...
       ACB ...
       ACC ...
  B
     BA ...
     BB ...
  C
     CA ...
     CB ...
     CC ...
```

Indented Representation

Figure 2. Functional Hierarchical Structure



Graphic Representation

```
SYSTEM I/O
  INPUT-A ...
  OUTPUT-B
     BA ...
     BB
       BBA ...
       BBB ...
       BBC ...
       (etc)
     BC ...
     (etc)
  INPUT-C ...
  OUTPUT-D ...
  (etc)
```

Indented Representation

Figure 3. I/O Hierarchical Structure

2.3.2 System Flow Relationships

System flow relationships can be shown by organizing the discrete requirements in terms of control flow (Figure 4) and information flow (Figure 5). As the functions of the system are defined, the control relationships between them can also be defined. These control relationships describe the logical order in which the system activities should be accomplished to satisfy the system mission and operational requirements. Conditions which determine the flow direction when two or more branches occur are also represented. Control-flow analysis provides a means of viewing the system from an activity-oriented perspective and is often referred to as functional-flow analysis. As a result of this analysis the requirements are viewed in a well-organized manner and missing or incomplete functions and relationships between the functions are identified. Final control-flow documentation becomes another effective means for communicating system requirements to design engineers.

On the other hand, the information-flow analysis (Figure 5) builds upon the I/O hierarchical structure (Figure 3) by providing a means of viewing the system as an information processing system. During this analysis the flow relationships between external system inputs and resulting outputs are identified. Quite often the most effective means of performing information-flow analysis is to trace an output back to system inputs, either external data, messages, or stimuli. As a result of this analysis the relationships between the associated functions and the internal information necessary to support the derivation of the output are identified.

Control-flow and information-flow analysis will identify necessary changes and additions to previously defined functions and constraints as well as to the hierarchial structures and other previously defined relationships. Missing or incomplete requirements can be determined and the deficiencies corrected.

Requirements engineering for systems which are primarily activity oriented may emphasize control-flow analysis over information-flow analysis. Other systems may be primarily information processing oriented and, therefore, the requirements engineering activities may concentrate more on information-flow analysis rather than control-flow analysis. In either case, both forms of analysis are involved in a total system definition.

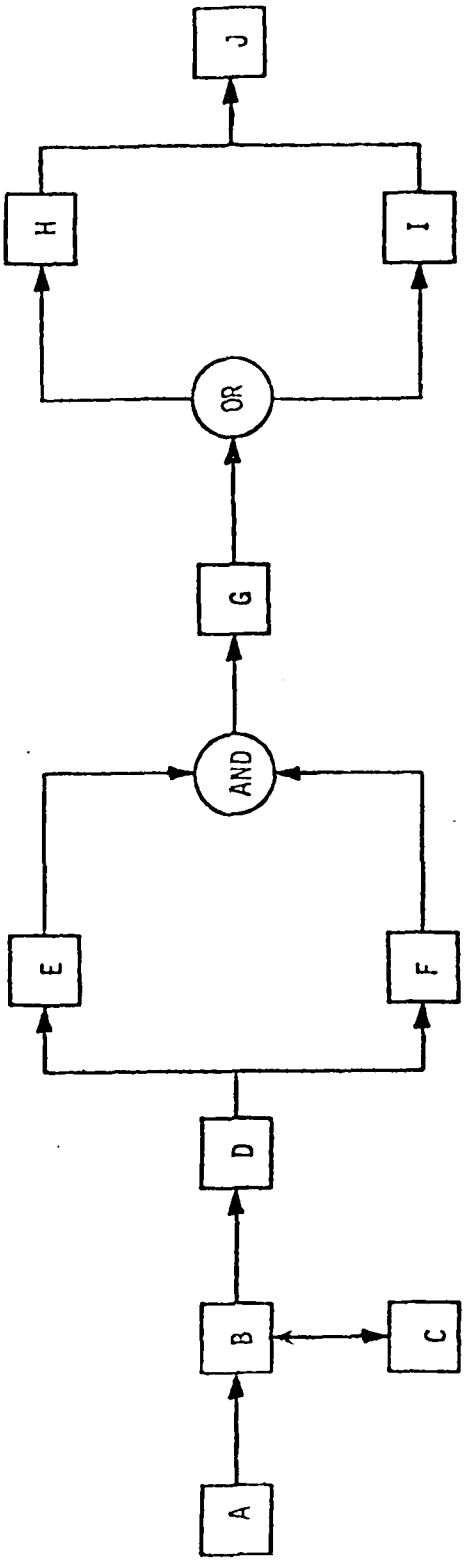


Figure 4. Control-Flow Diagram

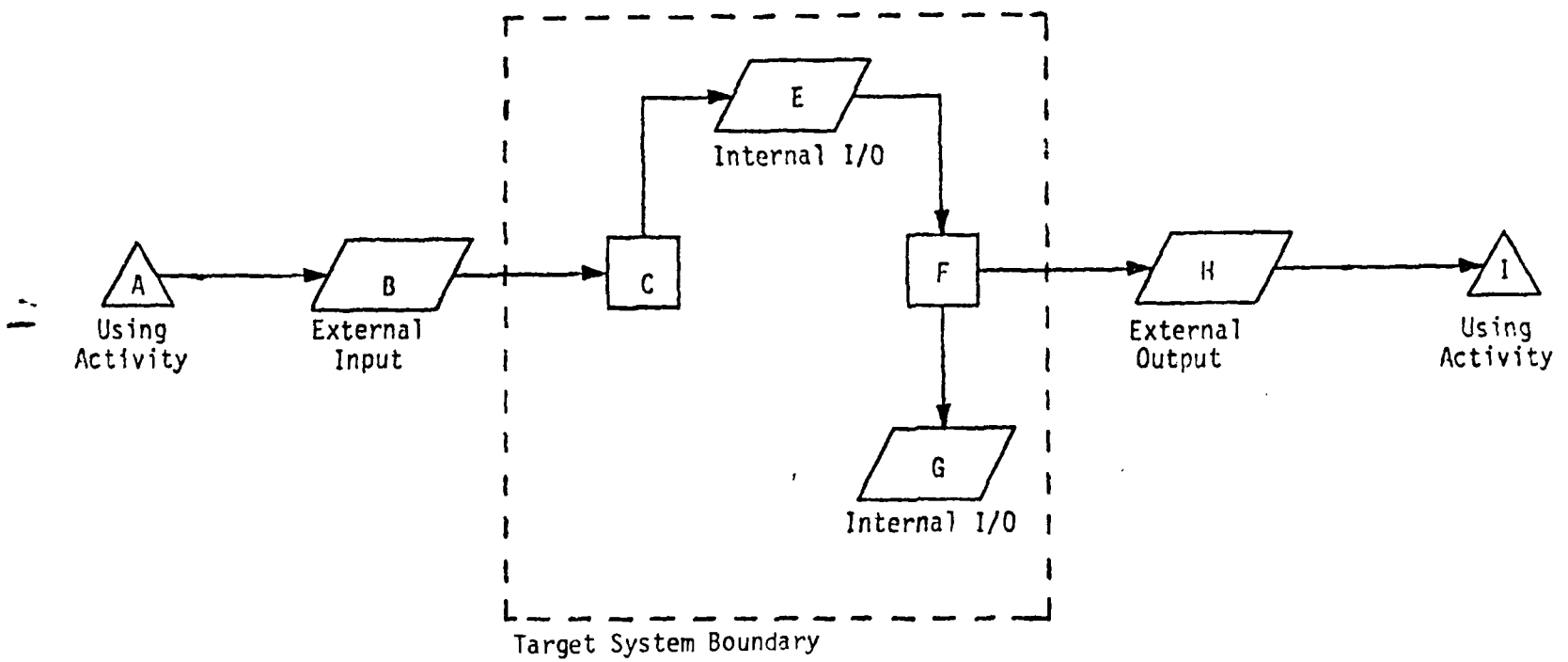


Figure 5. Information-Flow Diagram

2.3.3 Requirements Traceability Relationships

Identification of system traceability relationships is another effective means of identifying incomplete, unnecessary and missing requirements. During the requirements engineering activities, source documents are referenced for each requirement identified. Requirements traceability analysis provides the analyst with a means of verifying the requirements by linking each requirement to all forms of source documentation. These links, in the form of source references (sources), provide a link between the requirements from one set of system requirements (originating requirements) to the allocated requirements contained in the next level of the modeling or specification process using additional references (traces). For instance, in DoD Standard 7935.1-S the requirements may be traced from one higher level specification to another such as from the FD to SS, SS to PS etc., or in MIL-STD-490 from Type A to Type B specifications. Relationships can also be defined to other pertinent studies, analyses, and plans which are being accomplished concurrently with the requirements engineering activities, such as program management directives and plans, system sizing and timing studies, prototyping, simulations, test planning and the like. System test requirements (quality assurance), as well as subsequent test plans, procedures, and reports, can be effectively related to the functional specification (originating requirements). The links to associated system plans, analyses, and studies accomplished prior to, during and subsequent to the start of formal requirements engineering are crucial to the overall systems engineering concept. The traceability relationships also provide a bridge between requirements engineering activities and subsequent design and implementing activities, since the requirements can be traced from higher level functional (logical) specifications to design specifications, to product specifications, and to system test plans and procedures during the later phases of the system acquisition.

Throughout the system engineering activities, the analysts must be able to evaluate the impact of changes to the requirements. Whatever the reason (policy, economics, study or analysis results, new or modified requirements), the analyst must be in a position to determine the ramifications of changes to the system requirements as stated in various levels of specification. Once the area of impact is identified in the requirements engineering products (functional and I/O hierarchies, control and information flows, etc.) the traceability relationships defined during the previous requirements engineering activities (sources and traces) provide the capability to readily identify associated impacts to various parts of the system functional specification (logical model) and to trace the impacts to all other associated documentation: program directives, plans, studies and analyses, test plans and procedures, and allocated specifications (design

and product specifications). The impact can be readily analyzed and the appropriate actions taken.

2.4 Summary

Discrete and well-organized requirements support the primary goal of defining the operational needs of the using activity while giving the analyst visibility and control over the system functional definition (logical modeling) process. Discrete and well-organized requirements are prerequisities for the creation of good functional, design, and product specifications.

SECTION 3 STRUCTURED APPROACHES AND AUTOMATED TOOLS

3.1 Structured Approaches

In recent years a great amount of research and applications have been concentrated on techniques for defining, analyzing, and documenting the requirements for systems. Most of these techniques include the term 'structured' and are primarily manual techniques [14], [15], [16], [17], [18], [19], [20]. Some structured techniques do address the use of computer aids as a means of maintaining the requirements and producing documentation during the modeling process. The computer data bases are maintained either manually or by automated tools, and, are generally referred to as data dictionaries.

3.2 ISDOS Project - PSL/PSA

The University of Michigan's ISDOS Project began in 1968 to develop a more advanced computer-aided tool called the Problem Statement Language/ Problem Statement Analyzer (PSL/PSA). PSL/PSA is a more sophisticated data dictionary tool which provides the capability to record in an English-like language (PSL) the various objects (functions , inputs, outputs, etc.) of a system being defined (the target system) and relationships between the objects (hierarchy, flow, etc.). The objects and relationships are maintained in a computer data base called a requirements data base or PSL/PSA data base. The requirements data base can be used by the analyzer (PSA) to generate various reports about the target system such as hierarchical structures (functional and data), system flow (control and information) and many others.

Since the early 1970s, PSL/PSA has been applied by a large number of users with varying degrees of success. Although the tool is a very sophisticated software tool, there is no recommended approach for using it in developing logical models. This may be due in part to the research and development nature of the ISDOS Project or possibly a greater desire to make the tool a more general purpose product and thereby attract a larger number of PSL/PSA users. Many of the desired capabilities of the structured approaches being practiced today are not easily satisfied using PSL/PSA. Studies of PSL/PSA for large systems definitions have documented the need for improvements and some improvements have been incorporated in new versions [3],[4]. Some of these improvements have been made as part of the ISDOS Project and some have been made independently by various PSL/PSA users to meet specific modeling needs [5].

3.3 Utility of PSL/PSA

Various versions of PSL/PSA are currently being used in industry and in government agencies as indicated below.

(3.2) URL/URA (CARA OR CADSAT), an Air Force version of PSL/PSA, University of Michigan (ISDOS Project), 1974 [6] [7].

(3.2X) 3.2 plus extensions and modifications made by Logicon inc. for the Air Force, 1976 [5].

(4.2) PSL/PSA version available from the University of Michigan (ISDOS Project), 1977-1976 [8], [9], [10].

(5.1) Most recent version of PSL/PSA available from the University of Michigan (ISDOS Project), 1978-1979 [11], [12], [13].

Although new versions, specifically 5.1, do include some recommendations made in various studies and by the ISDOS users group, the basic core capabilities of the tool are available in the earlier releases such as the Air Force's version, URL/URA.

Learning the features of PSL/PSA as a software system alone (i.e., learning to use the language and reports) is in itself time consuming. The new user may become generally proficient in one to six months depending upon his experience with computer-aided techniques and the quality of available training and training documentation. Documentation on the tool, which varies in quality, is generally for reference purposes and the size and presentation of the material is most often perplexing to the new user. The new PSL/PSA user will often wonder why a language or report feature is available and also which ones are best suited to his needs.

Even after the tool features are reasonably understood, the new user must then determine the best approach to applying the tool to his requirements engineering problem. Since there are numerous structured approaches being expounded, the user must cope with the additional problem of determining which structured approach is best suited to his requirements engineering problem. If the new user is unfamiliar with any structured approach, he must become familiar with the new analysis techniques. The perplexities of the new user can be threefold: (1) learning PSL/PSA, (2) learning and/or determining which structured approach to apply, (3) and most importantly - defining and analyzing the target system.

3.4 Guidelines for Requirements Engineering Using PSL/PSA

This guidebook has been prepared to assist the new user of PSL/PSA in applying the tool in a productive manner to the logical modeling of a target system. This methodology was first documented as part of Logicon's final technical report for the RADC Requirements Standards Study (RSS) [3]. The RSS provided an approach to requirements definition and analysis and among other study results provided a description of the functional capabilities of automated requirements analysis tools.

The guidance provided in this guidebook is intended to aid the new as well as the experienced PSL/PSA user by providing a structured approach to the logical modeling process and to identify the current features of PSL/PSA which are best suited to the logical modeling activities described. The user is expected to use the references in appendices, especially those for the version of PSL/PSA being employed, for more detailed examples of the language and report features of PSL/PSA. The intent of this guidebook is to provide the framework for requirements engineering and an overview of the utility of PSL/PSA to support the logical modeling process.

SECTION 4 REQUIREMENTS ENGINEERING PROCEDURES

4.1 Introduction

The use of PSL/PSA must be restricted by a particular requirements engineering approach. A poorly defined or inadequate approach will result in costly and inadequate results. Conversely a well defined approach enhanced through utilization of selective features of a tool such as PSL/PSA will result in specifications of consistently higher quality. A well-defined approach and selected use PSL/PSA as presented in this guidebook is strongly advised.

Requirements engineering is the method used to derive and document quality system requirements. Requirements engineering as used in this guidebook is defined as follows:

> Requirements Engineering is an iterative process of defining the system requirements and analyzing the integrity of the requirements. This process involves all areas of system development preceding the actual design of the system. The products of the requirements engineering process can be evaluated for completeness, consistency, testability, and traceability. The essential goal of requirements engineering is to thoroughly evaluate the needs which the system must satisfy. Requirements engineering is principally concerned with the initial phases of the system acquisition life cycle; i.e., the definition phase.

Requirements engineering begins by identifying system boundaries and defining the system in terms of functional and constraint requirements. A functional requirement (function) is the statement of a need which must be fulfilled; a constraint requirement is a restriction on the function(s) which allows a solution to be derived. Constraints fall into one of the categories as illustrated in Table 1.

In performing requirements engineering, functions and their constraints as well as other requirements described in this section are extracted from source documents. These requirements can then be organized into hierarchical structures which reveal gaps which may be hidden by the overlapping and confusing statements of the original sources. Control and information-flows can also be explicitly defined to make the function-to-function and function-to-system data interactions visible. Finally, requirements can be traced from originating

Table 1. System Requirement Types

SYSTEM REQUIREMENTS	FUNCTIONAL REQUIREMENTS (functions)	The set of discrete functions which identify the pure design free or solution independent needs of the system as a whole. The functional requirements identify what must be accomplished while avoiding solution statements or overtones.	
	CONSTRAINT REQUIREMENTS (Constraints)	PERFORMANCE	How well the system functions must be accomplished, such as timeliness and accuracy. Also called performance characteristics, MIL-STD-490.
		PHYSICAL	Influences the design solution in a physical manner: power, size, weight, environment, human factors, existing system interfaces, GFP, etc. Also called Physical Characteristics, MIL-STD-490.
		OPERABILITY	Reliability, maintainability, availability, dependability.
		TEST	Identify the functional, performance, physical, operability, and design requirements which will be evaluated during system integration and test.
		DESIGN	The minimum or essential design and construction requirements which are a constraint on the functional requirements of the system during the design and construction of the system end-items (CIs/ CPCIs). Also called Design and Construction, MIL-STD-490.

source documents through various levels of system specifications to verify that all requirements have been allocated and implemented in the delivered system.

As stated in the definition above, requirements engineering is an iterative process of defining the system requirements and analyzing the integrity of the requirements for completeness, consistency, testability, and traceability (Figure 6). As the process continues the system requirements are defined and analyzed in a progressively expanding manner. The definition and analysis activities will move from one area of concentration to another as the results of previous activities reveal areas needing additional work. No single approach can be rigidly defined and applied which can take into account the many possibilities which must be considered. However, guidelines for requirements engineering and associated tasks can be defined and then tailored for specific requirements engineering applications.

This section presents a general framework for requirements engineering as illustrated in Figure 7 as well as recommended PSL/PSA language and report features which can be applied to each activity [BLOCK]. Each block represents a unique activity which can be accomplished in defining and analyzing system requirements. There is a continual interaction between the activities of each block, and although each block appears as a single activity, it is in fact part of a continuum. The selection of an actual approach for a given application is one of the tasks [BLOCK 2]. In a given application, not all blocks will necessarily be performed. The blocks selected must be responsive to the resources available to the project and the objectives of the analysis staff. The following is a brief description of each of the 15 blocks portrayed in Figure 7.

BLOCK 1 Identify and Review Source Documentation: The analysis team becomes familiar with the problem and all pertinent background information.

BLOCK 2 Produce Requirements Engineering Plan: A plan is developed to define the activities to be accomplished during BLOCKS 3-15; i.e., project schedule, tool features to be used, quality assurance provisions, etc.

BLOCK 3 Identify System Functions: System functions are identified in the source documentation and formally defined.

BLOCK 4 Organize Functions into a Hierarchical Structure: The functions in BLOCK 3 are organized so that each higher-level function is represented as an aggregate of more detailed functions.

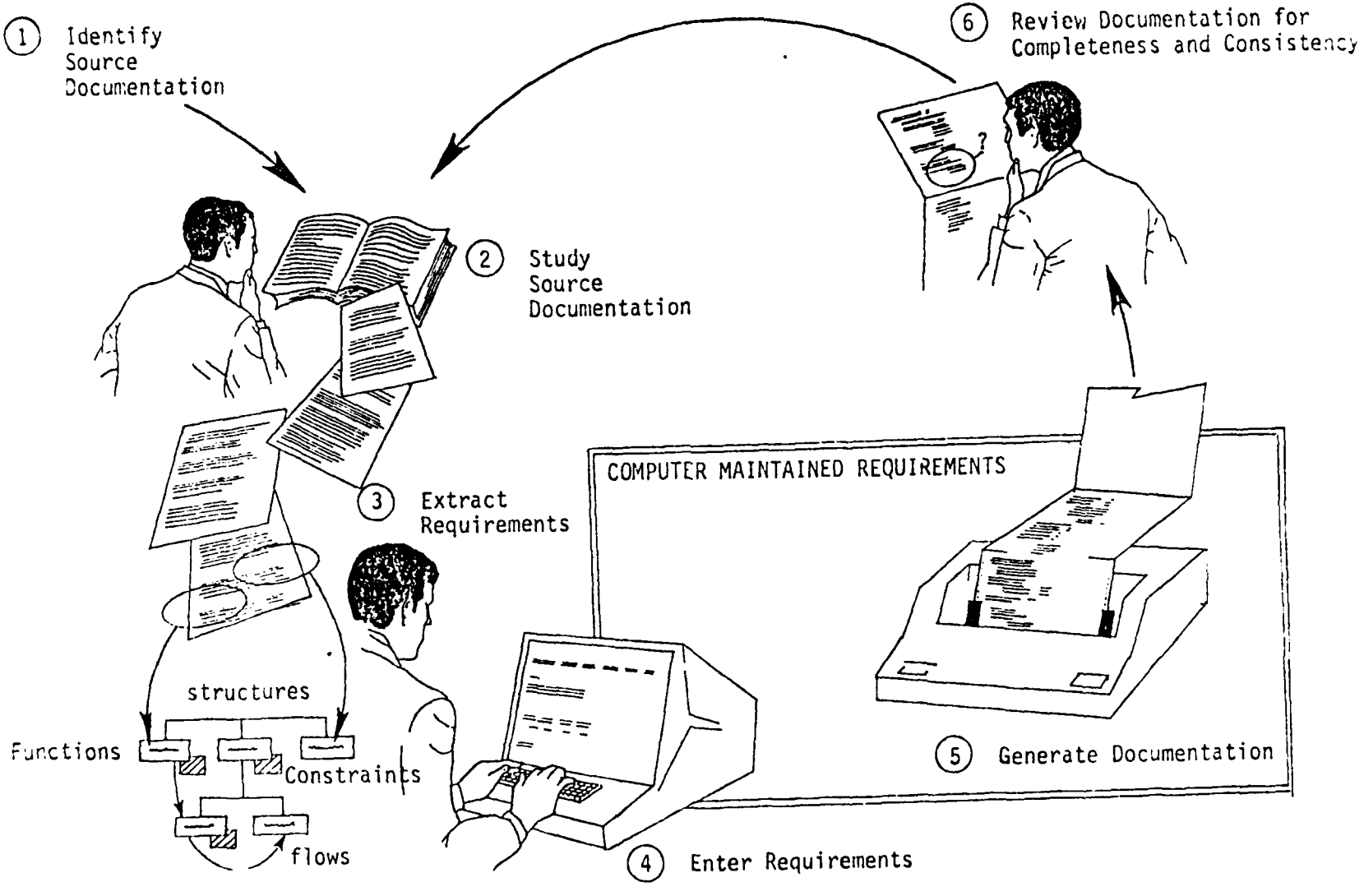


Figure 6. Requirements Definition and Analysis

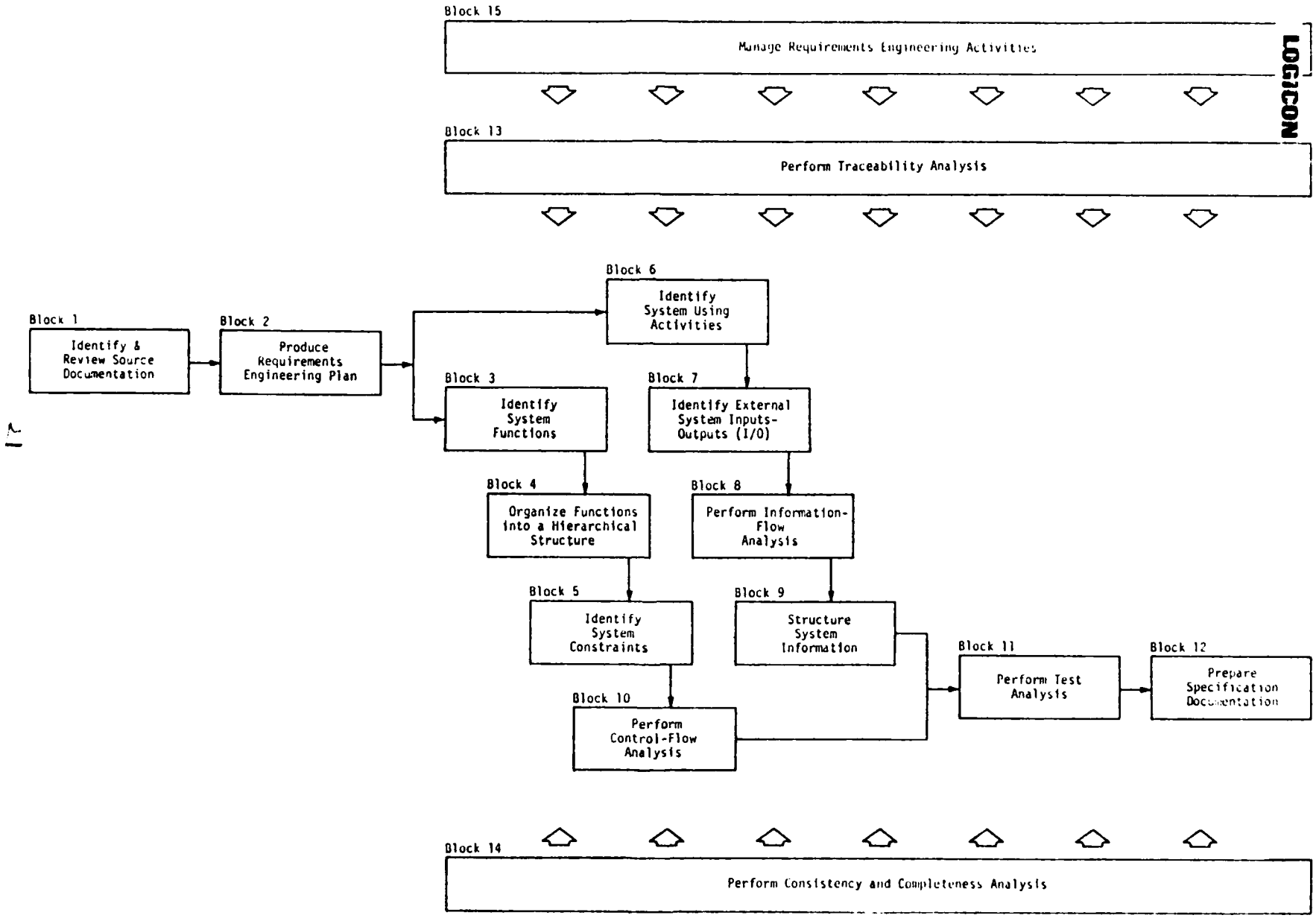


Figure 7. Requirements Engineering Procedures

BLOCK 5 Identify System Constraints: Constraints for the functions are defined where justified and attached to a specific function in the functional hierarchy.

BLOCK 6 Identify System Using Activities: System using activities (e.g., organizational units, external systems) which interact with the system from outside the system boundary are identified and structured hierarchically.

BLOCK 7 Identify External System Inputs-Outputs (I/O): Inputs and outputs to the target system are defined concurrently with the using activities.

BLOCK 8 Perform Information-Flow Analysis: Information flows showing the data flow between external inputs, target system functions, information within the system, and external outputs are defined.

BLOCK 9 Structure System Information: External and internal information is logically organized (i.e., data dictionary).

BLOCK 10 Perform Control-Flow Analysis: The sequences of system functions are defined as well as the control on the flow paths.

BLOCK 11 Perform Test Analysis: System requirements which will be subjected to formal testing and the test points in the control paths are determined.

BLOCK 12 Prepare Specification Documentation: Products of BLOCKS 3-11 are incorporated into specification documents such as DoD 7935.1-S, MIL-STD-490, or other approved formats.

BLOCK 13 Perform Traceability Analysis: Requirements are traced from one level model to another (or from one specification to another) to ensure that the subsequent models or specifications such as design and product specifications meet the users original needs.

BLOCK 14 Perform Consistency and Completeness Analysis: As errors are exposed in previous activities, the system description is refined by repeating BLOCKS 3-13 until a complete and consistent system definition has been achieved.

BLOCK 15 Manage Requirements Engineering Activities: During each of the preceding activities, project and technical managers determine progress by the analysis team and the status of the requirements data base.

In the following paragraphs each block in Figure 7 is explained in greater detail. Included in the description are the PSL/PSA tool features which have been selected as a subset of the language and analyzer capabilities best suited to the requirements engineering procedures in this guidebook. Alternate features may be selected based on the application needs and as experience is gained in application of PSL/PSA.

4.2 Identify and Review Source Documentation [BLOCK 1]

Source documentation as used in this guidebook includes all recorded information on a system such as:

- o planning and user requirements documents
- o specifications for similar systems, for system interfaces, and for existing or previously defined subsystems
- o documentation derived from engineering studies and prototyping or experimental test systems
- o user interviews and associated documentation

During this task the requirements engineering team shall individually review the source documentation in order to become familiar with the overall system requirements. During review sessions with other team engineers, the analysts shall perform a general evaluation of the requirement types (objects) contained in the source documentation. The review of the source documentaton and the assessment of requirement types are prerequisites for developing the requirements engineering plan [BLOCK 2]. As the analysis activities [BLOCKS 3-14] continues, additional source documents will be identified and evaluated.

4.3 Produce Requirements Engineering Plan [BLOCK 2]

After review of the source documentation, the analysis team shall determine the specific approach to accomplishing BLOCKS 3-15. This approach shall take into account all available resources including personnel, schedule, and financial considerations. The planning shall detail the methodology to be applied (tools, techniques, conventions, etc.), specific tasks to be accomplished, personnel assignments, resource descriptions, schedules and milestones, preliminary and final documentation to be produced [BLOCK 12], progress reviews and quality assurance and status reporting procedures. The results shall be described

in a requirements engineering plan which will be updated throughout the analysis activities to reflect necessary changes. The requirements engineering plan serves to define the objectives and means of the analysis to be performed and assists project staff in performing the tasks and meeting the goals of the project. New analysts and other uniformed observers will find the requirements engineering plan useful for becoming familar with the requirements engineering project throughout the project's life.

PSL/PSA language and analyzer features shall be determined by review of BLOCKS 3-15, Appendices B and C, and the language and analyzer references in Appendix A. This selection shall insure that the analysis proceeds in a uniform manner, and the PSL/PSA features satisfy the requirements engineering project objectives. The objective is not to build and maintain requirements data bases using PSL/PSA, but to define, analyze, and document quality system requirements using the best features of PSL/PSA which satisfy the analysis activities defined in this guidebook. Therefore, PSL/PSA shall be used as conservatively as possible to achieve the objectives of the project.

4.4 Identify System Functions (BLOCK 3)

During this task the source documentation is analyzed and the system functions (PROCESSES), necessary to control or produce the desired outputs from the available inputs, shall be identified. A function is a discrete activity within a system. The collection of discrete functions, defines the total activities which must be accomplished by the system to achieve a given objective. The functions identified shall range from high level (first possible functional breakout of the system) to detailed lower level functions (functional primitives) which represent finite, distinct actions to be performed by system equipment, computer programs, personnel, facilities, procedural data, or combinations thereof.

Naming a function is an important part of the requirements engineering process. The following conventions for developing function names shall be applied:

- Each function shall be given a unique name conforming to the function name of the sources or its characteristics.
- The function name shall be succinct. This increases the ability of the reader to retain the idea being expressed, especially for large or complex systems consisting of many functions.

o The function name shall not imply any preference for a design solution, even if the source of the requirement specifies some aspect of design.

o When a function name exceeds 30 characters, it can be reduced by abbreviating parts of the name. Since the tool does not have a means of recording abbreviations used in a name, a separate glossary must be maintained. Every attempt shall be made to avoid abbreviations, since they decrease the readability of the name especially for those unfamiliar with the abbreviations employed. The need to abbreviate is often a sign of an ill-defined requirement or a combination of requirement types and/or modifiers.

o Functions which are primitives shall include a PROCEDURE statement. PROCEDURE statements shall include any combination of the following: (1) Structured English stating the logical steps which represent the function, (2) Decision Tables, or (3) Decision Trees [15], [19].

o Function names shall conform to the following constructs:

CONSTRUCT	EXAMPLE
Verb-Object*	assemble-requisition asem-requisition-for-publisher
Compound-Verb-Object*	prepare-and-distrib-reports

* with or without modifiers, such as adjectives and/or preprepositional phrases.

4.5 Organize Functions into a Hierarchical Structure [BLOCK 4]

In conjunction with identifying the system functions as described in BLOCK 3, the functions shall be arranged into logical hierarchical structures (Figure 2). This form of organization is suited for structuring system functional requirements in a logical arrangement for communicating system functions and the hierarchical relationships between the functions to design engineers. The functional hierarchy provides a view of the system as an aggregate of functions broken down into a logical arrangement of subordinate discrete activities which must be performed. The sum of the activities of the functions on a given level are equal to the activity at the next higher level in the hierarchy. This principle means the total system activities are defined by the functions at the lowest level in the hierarchy (functional primitives). This logical form of organization is

distinguished from information-flows [BLOCK 8] and control-flows [BLOCK 10].

The functions of the system shall be grouped into higher levels of organization representing the first possible breakout of the system. Upper-level functions shall be refined by the identification of subordinate levels. Each level of the hierarchy shall be limited to 2-7 functions. The limit of seven functions has been shown to increase the human understanding of the system functional requirements. Should the need exist for more than seven functions at a given level, the analysis team shall review upper levels of the hierarchical structure and make any adjustments that can be made to resolve the problem.

During the course of the organization of functions into a logical hierarchy, the names of previously defined functions may be altered in order to conform to the logical structuring at lower levels. On the other hand, the logical structuring may necessitate the creation of pseudo-function names in order to provide a means of organizing functions under special and meaningful groupings. In addition, the hierarchical structuring may necessitate identification or creation of new functions which were omitted during previous analysis.

If the functional hierarchy is derived from leveled data-flow diagrams [BLOCK 8], the functional hierarchy can be derived as a result of the hierarchy of parent-child relationships. This method is preferred over other less structured means of functional decomposition, because it provides a balance between the decomposition of functions in parallel with the decomposition of data flow and data structure.

The language statement which allows a function (PROCESS) to be hierarchically related to another is the PART/SUBPART relationship within the PROCESS section. The reports which best shows the functional hierarchy is the Structure Report. Other reports display only immediate PART/SUBPART relationships: Picture Report (with the structure option in effect) and the Formatted Problem Statement Report.

4.6 Identify System Constraints [BLOCK 5]

In conjunction with the identification of system functions and organizing functions into a hierarchical structure, the analysis team shall identify all system constraints. The constraint requirements shall be limited to performance, physical, operability, and design. Test requirements are addressed in BLOCK 11. Constraint requirements shall be derived from available source documentation or from the results of trade-off studies, feasibility studies or advanced development studies. Each constraint requirement shall be related to specific function

levels in the functional hierarchy [BLOCK 4]. A constraint applied to a given level in the functional hierarchy implies that the constraint is applicable to each lower level function in the hierarchy. As the constraint analysis continues the constraints may be more specifically allocated to lower level functions in the functional hierarchy. Constraints which are not clearly justified from available documentation shall be eliminated from consideration until documented justification is available. All constraint requirements shall be stated in specific, quantifiable parameters, either as a single value or range of values, including the unit of measure, limits, accuracy or precision, and frequency.

During the course of identifying the various constraints imposed on the functions of the system, the analysis team shall verify that no combination of constraints results in excessive or unrealistic requirements [BLOCK 14]. Technical risks identified by the analysis of constraints shall be followed up by additional studies to resolve areas of conflict. References to any source documentation or analysis and studies which support and justify each constraint requirement shall be maintained using SOURCES [BLOCK 13].

Although some methodologies for requirements engineering emphasize deferring the identification of constraints over preference for data-flow analysis [BLOCK 8], it is better to maintain records of the constraints as they are identified. This allows the analyst a means of deferring the constraints in a way which assures that each one will not be lost or forgotten and also provides the basis from which constraint analysis may proceed.

There are several means of identifying constraints using the language. One way is to use the ATTRIBUTE and ATTRIBUTE-VALUE statements. For example, the following statement shows a performance (pf-) constraint associated with the function "validate-time-cards".

```
PROCESS validate-time-cards;
ATTRIBUTES ARE  pf-1-time-per-week  performance-constraint;
```

The meaning of the ATTRIBUTE can be expanded through the use of the DESCRIPTION statement and SOURCE statement in the ATTRIBUTE section. DESCRIPTION allows elaboration on the meaning of the constraint (i.e., pf-1-time-per-week) by allowing text (comment-entries) to be entered. SOURCE provides the means of identifying the source of the constraint requirement (analysis, studies, reports, etc.) and therefore provides for traceability [BLOCK 13]. To expand the example, the following DESCRIPTION and SOURCE statements could be provided:

```
PROCESS  validate-time-cards;
ATTRIBUTES ARE   pf-1-time-per-week  performance-constraint;

DEFINE  pf-1-time-per-week  attribute;
DESCRIPTION;
----------------------------
-----comment-entry-----------
----------------------------;
SOURCE  source-identifier;
```

In this example the first two statements define the function and the associated constraint. The remaining statements expand upon the constraint by renaming it and adding descriptive text and the source of the constraint. The use of the prefix (pf-) for performance in the ATTRIBUTE name appears to be redundant with the ATTRIBUTE-VALUE. However, the use of the prefix makes other reports more useful, since many of the reports are sorted on object names. For example, the Name List Report will group all performance constraints together where the prefix "pf-" is used in the ATTRIBUTE name.

The ATTRIBUTE and ATTRIBUTE-VALUE can be used for other contraint types. Prefixes for all constraints are as follows:

performance	pf-
physical	py-
design	dn-
operability	op-

The Attribute Report and Formatted Problem Statement Report will display the constraints (ATTRIBUTES) along with the functions (PROCESSES) which they contrain.

Another way to express constraints is the use of MEMOS. Using the same example, the following illustrates the application of MEMO as a constraint:

```
PROCESS  validate-time-card;
SEE-MEMO  pf-1-time-per-week;

MEMO  pf-1-time-per-week;
DESCRIPTION;
----------------------------
-----comment entry-----------
----------------------------;
SOURCE  source-identifier(s);
```

The DESCRIPTION and SOURCE statements for the MEMO achieve the same results as the ATTRIBUTES section. Here the prefix is distinguish the MEMO as a constraint and again are useful for producing and using other report outputs. The Structure Report (3.2X) and Formatted Problem Statement Report will display the constraints (MEMOs) along with the functions (PROCESSES) which they constrain.

Another means for defining performance constraints which involves rates is to use the HAPPENS statement. The HAPPENS statement specifies the number of occurrences of a function (PROCESS) in a given time interval. The following statements illustrate the HAPPENS statement to express the previous performance constraint:

```
PROCESS  validate-time-card;
HAPPENS  one  TIMES-PER  week;
```

The Frequency Report and Formatted Problem Statement Report can be used to display this form of performance constraint. Other contraint types (-py, -op, -dn) have to be represented using ATTRIBUTES or MEMOS.

4.7 Identify System Using Activities [BLOCK 6]

Using activities (organizations, operational units, or operator positions) which interact with the target system shall be identified. The identification of using acitivities provides the basis of information-flow analysis [BLOCK 8]. The identification shall include the names of using organizations described in the sources or through other determinations such as human engineering studies which lead to the identification of using activities. Lower level position names, such as specific operator positions, shall be identified and described to the level of detail required for the associated functions.

Using the INTERFACE object and PART/SUBPART statements, the requirements engineer can define and structure all using activities which interact with the target system. Identification of using activities is best accomplished in conjunction with information-flow analysis [BLOCK 8]. As INTERFACES are identified (i.e., external activities which are origins or destinations of targer system products), language statements should be prepared. The hierarchical structuring of INTERFACES using the PART/SUBPART relationships is generally not required unless this information supports the understanding of the target system. An example entry for a using activity is as follows:

```
INTERFACE   keypunch-operator;
PART        data-processing-department;
RECEIVES    keypunch-forms;
GENERATES   keypunch-deck;
SOURCE      source-identifier(s);
```

The structure of the using activities (INTERFACES) is provided by the Structure Report if the PART/SUBPART relationships have been entered. The Formatted Problem Statement Report, Extended Picture Report, and Picture Report display INTERFACES with varying degrees of informational value.

Another approach to identifying and structuring using activities is to treat them as PROCESSES. The using activity is defined as a PROCESS and given a name which is in the noun form (e.g. data-processing-department). The hierarchy of using activites can be maintained using PART/SUBPART statements under the PROCESSES. In this case one major branch of the process structure can be dedicated to the using activities and another branch set aside for the target system functions. This approach still allows identification of using activities (although not specifically by language type, INTERFACES), allows hierarchical structuring, and data flow. Data flow becomes more simplified with this approach because without the use of INTERFACE, the use of INPUTS and OUTPUTS are no longer practical. The only practical collections of data remaining are SETS, ENTITIES, GROUPS/ELEMENTS. This restriction is not considered severe. To beginning users, this approach has been found to simplify the application while still effectively addressing the needs of the requirements definition and analysis process. The tool allows the user to alter the type of the object from PROCESS to INTERFACE should it be decided at a later date that it is desirable to do so. However, because of the uniquenesses of each language object, a change in the type of one object can necessitate other changes within the object being changed as well as other objects. Therefore, it is suggested that use of PROCESS as a using activity should be adhered to throughout the project once it is decided to do so.

4.8 Identify External System Inputs-Outputs (BLOCK 7)

In conjunction with identifying the using activities, the analysis team shall identify the output (responses) required from the system. Output information consists of reports needed by using activities as well as system messages necessary for the operation, maintenance, and control of the system. Subsequent to each output being defined, the associated system inputs (stimuli) shall be identified. Each input or output (I/O) shall be given a unique name conforming to the I/O name in the sources or its characteristics. Parts of an input or output shall be identified and named as the requirements engineering process continues

[BLOCK 9]. References to source documentation (SOURCES) which identify the need for the I/O shall be maintained [BLOCK 13]. Each I/O shall be suplemented by a brief description of the I/O and its purpose.

The INPUT and OUTPUT language objects are designed to be used as their names imply, that is, external inputs and outputs of the target system. These collections of information can be used to express physical or logical collections of data. As physical collections they are thougnt of as containers of data values which consist of GROUPS/ELEMENTS. INPUTS and OUTPUTS may also be contained in larger collections of information, INPUTS, OUTPUTS, and SETS. For requirements engineering purposes, the logical collections are the preferred means of using the language.

Example INPUT and OUTPUT statements are as follows:

```
INPUT  time-card;
GENERATED BY  keypunch-operator;
DESCRIPTION;
------------------------------
-----comment entry------------
------------------------------;
SOURCE  source-identifier(s);

OUTPUT  keypunch-forms;
RECEIVED BY  keypunch-operator;
DESCRIPTION;
------------------------------
-----comment entry------------
------------------------------;
SOURCE  source-identifier(s);
```

4.9 Perform Information-Flow Analysis [BLOCK 8]

During this analysis, the flow relationships between external system inputs and resulting outputs shall be identified in information-flow diagrams (also called data flow diagrams and data flow graphs). These diagrams provide the basis for determining that each I/O is used, derived, or updated. An effective means of information-flow analysis is to trace an output back to the system input: external data, messages, or stimuli. This method permits the relationships between associated functions and the internal information necessary to support the derivation of the output to be identified. Structured approaches for information-flow analysis are described by DeMarco [15], Gane and Sarson [19], and Ross [14]. The data flow within the target system boundary can be described using the

following language relationships as illustrated in Figure 8:

USES — This relationship indicates that a function (PROCESS) on the path uses external information (INPUT*) or internal system information (ENTITY*) as an input to its activities.

DERIVES — This relationship indicates that a function (PROCESS) on the path derives either external information (OUTPUT*) or internal system information (ENTITY*) as a product of its activities.

UPDATES — This relationship indicates that a function (PROCESS) on the path updates internal system information (ENTITY*) as part of its activities.

* or their higher/lower level collections, i.e., SETS, GROUPS/ELEMENTS

Compound forms can also be used such as USES- TO DERIVE and USES-TO UPDATE. These forms are better since they result in a more accurate definition of the function's transaction and result in more meaningful PSA reports, i.e., Picture and Extended Picture reports.

The information flow shall indicate the relationship between system functions and system information (external and internal system I/O) and shall not imply any lapse in time, or intermediate I/O be used, derived, or updated.

For the purpose of information-flow analysis the external using activities identified during BLOCK 6 are integral to the definition of the system total information flow since the using activities are the origins and destinations of system external I/O. The flow across the target system boundary (between INTERFACES and PROCESSES, if INTERFACES are being used to represent using activities) shall be described using the following information-flow relationships as illustrated in Figure 8:

GENERATES — This relationship indicates (1) a using activity (INTERFACE) is the origin of the external input (INPUT) or (2) a function (PROCESS) is the origin of the external output (OUTPUT).

RECEIVES — This relationship indicates (1) a function (PROCESS) is the recipient of the external input (INPUT) or (2) a using activity (INTERFACE) is the recipient of the external output (OUTPUT).

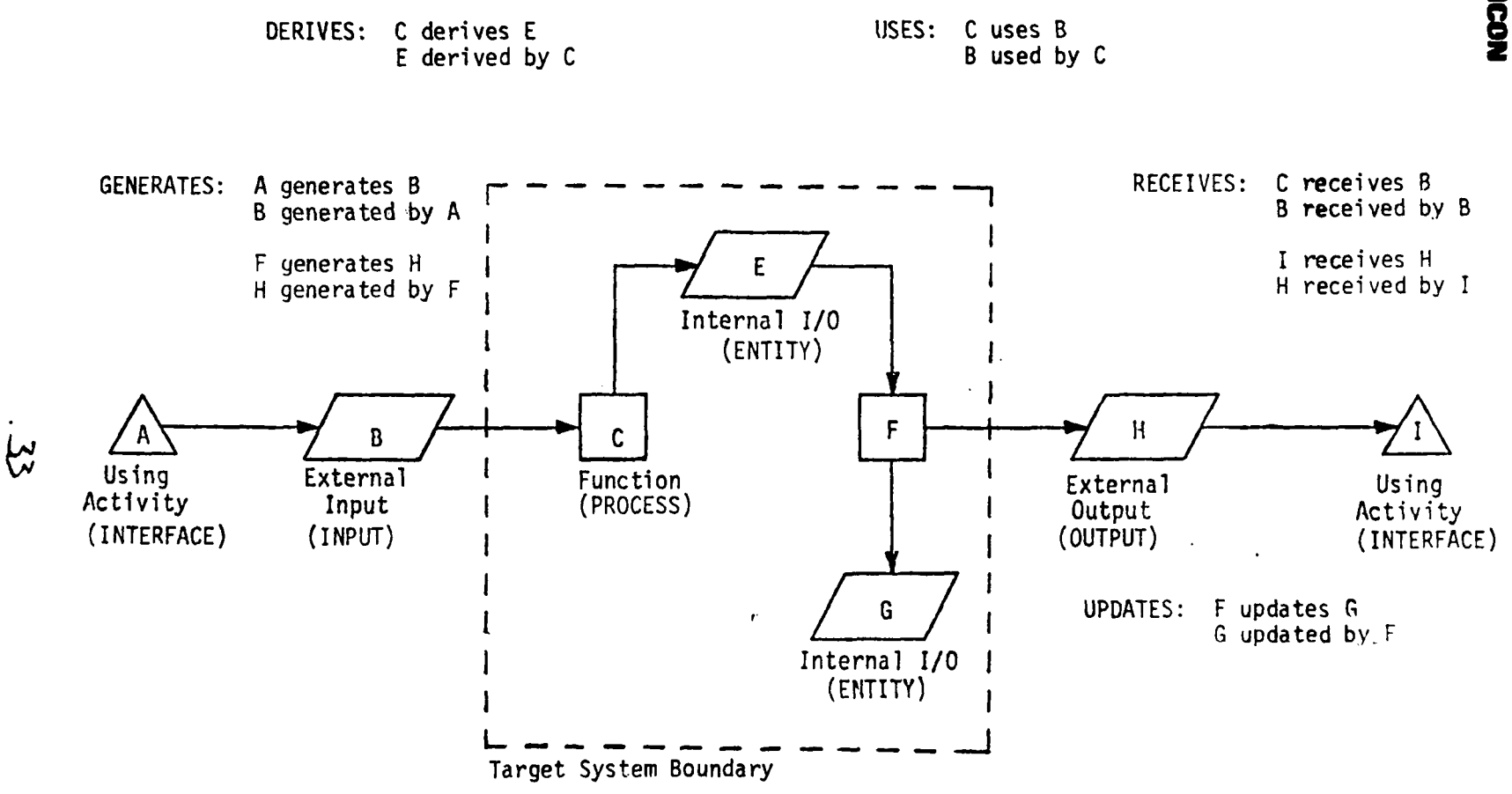


Figure 8. Information-Flow Diagram with PSL constructs

Two possible sets of language statements which corresponds to the information-flow diagram in Figure 8 are shown below. Each set of statements should be supplemented by a SOURCE statement where it is appropriate to maintain traceability between the flow requirement and sources of the requirement (BLOCK 15).

SIMPLEX FORM

```
INTERFACE a;
GENERATES b;

INPUT b;
USED BY c;

PROCESS c;
RECEIVES b;
DERIVES e;


ENTITY e;
USED BY f;

PROCESS f;
UPDATES g;
DERIVES h;
GENERATES h;

OUTPUT h;
RECEIVED BY i;

INTERFACE i;
```

COMPOUND FORM

```
INTERFACE a;
GENERATES b;

INPUT b;

PROCESS c;
RECEIVES b;
USES b TO DERIVE e;

ENTITY e;

PROCESS f;
USES e TO DERIVE h;
USES e TO UPDATE g;
GENERATES h;

OUTPUT h;

INTERFACE i;
RECEIVES h;
```

Reports which present the information flow are the Data Process or Data Process Interaction Report, Element Process Analysis Report and Element Process Usage Report, Extended Picture Report, Formatted Problem Statement Report, Function Flow Data Diagram Report, Picture Report, Process Input/Output or Process Summary Report, and the Structure Report. None of these reports present leveled data flows.

4.10 Structure System Information (BLOCK 9)

During BLOCK 7, the external I/O (INPUTS, OUTPUTS) and their superordinate and subordinate parts are defined. During BLOCK 8, the internal I/O (ENTITIES) and their superordinate and subordinate parts are defined as the information-flow analysis is performed. During this activity, the external and internal

information is arranged into logical hierarchical structures (data structures) as illustrated in Figure 3. The emphasis on the data structures is to organize the I/O and their superordinate and subordinate parts into logical organizations or simply as groupings of information. Structuring the I/O is an effective means of identifying incomplete or missing I/O and for communicating the input and output requirements to design engineers.

Parts of I/O identified during BLOCK 7 and 8 shall be associated with other I/O and organized into hierarchical structures. Changes and additions to the I/O hierarchical structures may be required as information-flow analysis (BLOCK 8) is accomplished. The upper parts of the individual I/O hierarchical structures shall be equivalent to the aggregate of the subordinate parts in the hierarchy. During the course of organizing the I/O into a hierarchy, the names of previously defined I/O may be altered in order to conform to the data structure being defined. On the other hand, the structuring may necessitate the creation of pseudo input/output names in order to provide an effective means of organizing the data structures in special and meaningful groupings. In addition, the hierarchical structuring may necessitate the identification of additional I/O requirements which were omitted during earlier requirements engineering activities.

The identification of the I/O, its description, structure, and other features is called a data dictionary. The following example describes a typical data dictionary entry for an INPUT. Each set of statements can be supplemented by a SOURCE statement where traceability is desired or required (BLOCK 13).

```
INPUT  time-card;
CONSIST OF  employee-name, employee-no, week-ending-date,
            project-numbers, mon-hrs, tue-hrs, wed-hrs,
            thr-hrs, fri-hrs, sat-hrs, sun-hrs,
            total-hrs-by-project-no, total-hrs-by-week-day;
DESCRIPTION;
     A card used by the employee to record weekly hours
     against projects throughout the work week;
SYNONYM  employee-time-card;

GROUP  employee-name;
CONSISTS OF  last-name, first-name, middle-initial;

ELEMENT  last-name, first-name, middle-initial;
```

Several reports will present various aspects of the data dictionary. The Contents Report provides the data structure in an indented format beginning with the object name specified (SET, INPUT, OUTPUT, ENTITY) down to the lowest level defined using the

CONSISTS/CONTAINED statements. The Structure Report (version 5.1, with the appropriate options in effect) provides an indented hierarchy of the SETS, INPUTS, OUTPUTS, and ENTITIES based on the PART/SUBPART and SUBSET/SUBSETS relationships. The Formatted Problem Statement Report provides the same data structure (CONSISTS/CONTAINED) information as well as DESCRIPTIONS, SYNONYMS, and SOURCES. Other reports which aid in development and analysis of data structure and contents are identified in Appendix C.

4.11 Perform Control-Flow Analysis (BLOCK 10)

Control-flow analysis provides a means of viewing the system from an activity-oriented perspective and is often referred to as functional-flow analysis. Control-flow diagrams like Figure 9 describe the sequential flow between system functions. Whereas the information flows do not indicate any preferred ordering of functions, control flows describe the order in which functions are to be activiated. In many applications these control paths become meaningful in the understanding of the system and are desired by the target system user or documentation requirements. Where control-flow analysis is desired or required, it is recommended that the ordering of functions be prepared after completion of information-flow analysis.

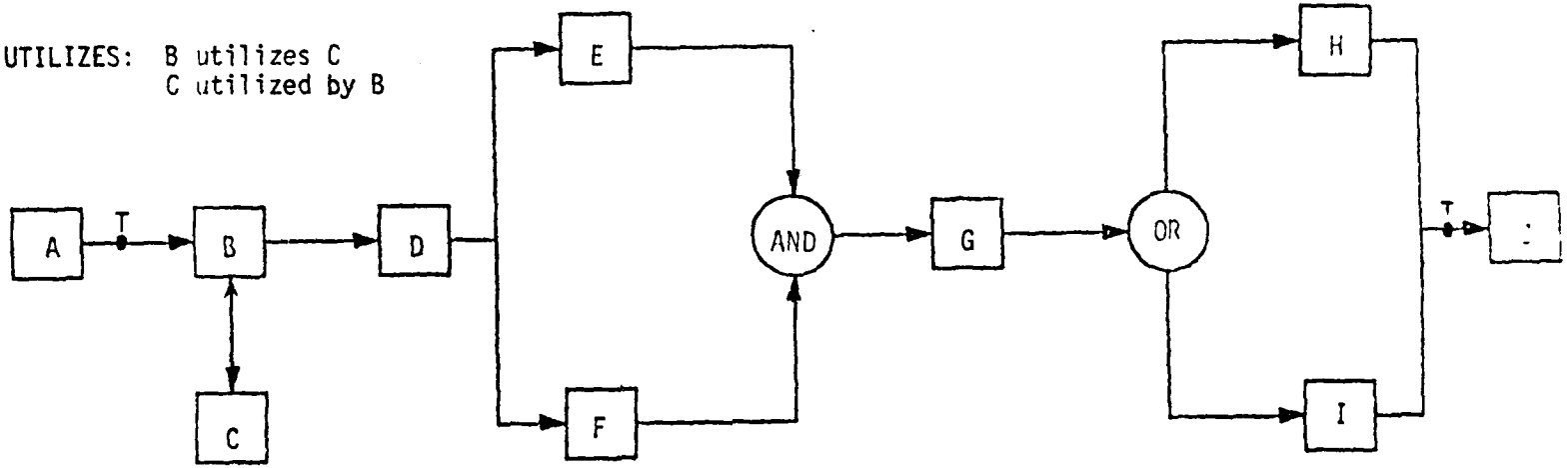
Control-flow diagrams, like information-flow diagrams, shall indicate only the relationship between system functions and shall not imply any lapse in time, or intermediate activity. The sequence of functions (PROCESSES) and conditions which determine the flow directions shall be described using the following control-flow relationships as illustrated in Figure 9:

TRIGGERS — This is a sequential relationship between two or more functions (PROCESSES).

UTILIZES — This relationship indicates that a function (PROCESS) on a path is dependent upon the use of one or more other functions in order to accomplish its activities. A single function or sequence of functions may be defined once and utilized as frequently as necessary in the control flow without having to be redefined for each use.

CONDITION — AND condition: Functions (PROCESSES) preceding the AND must be accomplished before the flow can continue.

OR condition: Any one of the alternate paths may lead to the next function (PROCESS).



TRIGGERS: A triggers B
B triggered by A

G triggers H, I
I triggered by G
H triggered by G

etc.

CONDITION: G triggered when condition-name BECOMES (TRUE
(AND) (FALSE

CONDITION: H triggered when condition-name BECOMES (TRUE
(OR) (FALSE

I triggered when condition-name BECOMES (TRUE
(FALSE

T = Test Point (EVENT)

Figure 9. Control-Flow Diagram with PSL constructs

The use of the TRIGGERS and UTILIZES relationships provides the basic sequence of functions (functional flow). The addition of CONDITIONS statements modifies the functional flow by providing control descriptions on diverging or converging flow paths and thereby makes the sequence of functions a control flow.

The control flow shall be restricted to concepts backed by system engineering studies or the like which clearly resolve any uncertainty of technical risks associated with the flow concept described. Where uncertainty exists the relationships shall be described as tentative or not completed, as appropriate, until subsequent analysis resolves the uncertainty. As the control flow is identified, SOURCES of the control requirements (studies, reports, etc.) shall be maintained [BLOCK 13].

One possible set of language statements which corresponds to the control-flow diagram in Figure 9 is shown below. Each set of statements should be supplemented by a SOURCE statement where it is appropriate to maintain traceability between the flow requirement and sources of the requirements [BLOCK 13].

```
PROCESS a;
TRIGGERS b;

PROCESS b;
UTILIZES c;
TRIGGERS d;

PROCESS d;
TRIGGERS e,f;

PROCESS g;
TRIGGERED BY e,f;
TRIGGERED WHEN  condition-name  BECOMES TRUE;
TRIGGERS h,i;

PROCESS h;
TRIGGERED WHEN  condition-name  BECOMES TRUE;
TRIGGERS j;

PROCESS I;
TRIGGERED WHEN  condition-name  BECOMES FALSE;
TRIGGERS j;
```

In the above example, "condition-name" is the object name (maximum of 30 characters) called a CONDITION (AND condition, and OR condition). The condition-name is a statement which represents a condition which can be either in a true or false state. Once the condition-name is determined, the TRUE or FALSE is selected as appropriate to the logic of the flow being defined.

Several reports display control-flow information. The Process Chain Report provides a graphic output showing the chain of TRIGGERS, UTILIZES (5.1 only), and CONDITIONS (5.1 only). All control-flow relationships are shown in the Formatted Problem Statement Report and also in the Structure Report (3.2X only). The Dynamic Interaction Report (5.1 only), Picture, and Utilization Analysis Report (5.1 only) also show various aspects of control flow.

4.12 Perform Test Analysis [BLOCK 11]

Test requirements identify the system requirements which will be evaluated during system integration and test. The principle objective of test analysis is to identify which areas in the system definition shall undergo formal test and verification. This is achieved by identifying test points on the control-flow paths (Figure 9). Test points shall be added to the flow paths at the selected test data sampling locations as the control flow is developed. The selection of test points shall be accomplished concurrent with the test planning activities.

The association between system test plans, analyses, and studies documented prior to, during, and subsequent to the start of formal requirements engineering is crucial to the overall requirements engineering concept. Documented test objectives preceding formal requirements engineering shall be analyzed. As a result, test points in the control flows shall be selected which provide data for various test cases and support testing objectives. References (SOURCES) shall be maintained between the test points and associated test plans and other supporting documentation [BLOCK 13].

The language has no direct means of representing test points on the information and control flows. However, one means of representing these test points can be achieved through the use of the EVENT object. In this case the EVENT denotes a point on the flow where test (validation) data is desired during the system testing. The procedures which will be used to analyze the collected data can be described in the DESCRIPTION. References (SOURCES) to test plans and procedures can be identified. A single test case may be defined as an EVENT made up of several test points (EVENTS) using PART/SUBPART relationships. All test cases can be structured to provide a comprehensive hierarchy of the system/subsystem and unit level testing which will be performed during system integration and test. The following statements illustrate a test case for the control flow shown in Figure 9:

```
EVENT  test-case-name;
SUBPARTS ARE  test-point-a, test-point-b;
```

```
DESCRIPTION;
     comment entries describing the procedures for
     analyzing the test point data.  This may be
     extracted from or entered into test plans and
     procedures as they are developed;
SOURCE  source-identifier(s);


EVENT  test-point-a;
CAUSED BY  a;
TRIGGERS  b;
DESCRIPTION;
     comment entries describing the test data to be
     collected;
SOURCE  source-identifier(s);


EVENT  test-point-b;
CAUSED BY  h,i;
TRIGGERS  j;
DESCRIPTION;
     comment entries describing the test data to be
     collected;
SOURCE  source-identifier(s);
```

The PART/SUBPART relationships are available only in version 5.1.

The test cases and test points (EVENTS) will appear in numerous reports depending on the version available. The Structure Report (5.1) will display the hierarchy of the test cases/points based on the PART/SUBPART relationships (5.1). The Process Chain Report (all versions) and the Dynamic Interaction Report (5.1) will display the test points (EVENTS) as part of the functional/control flow. The Formatted Problem Statement Report generated for EVENTS will provide a complete display of the test cases/points. The lack of PART/SUBPART relationships in version 3.2 and 4.2 can be worked around by the analysis team using an alphanumeric naming convention for the test case/points (EVENT names) which will allow reports such as the Formatted Problem Statement Report to output ordered by the name of the EVENTS. In this case the Name-Gen (3.2) or Name-Selection (4.2, and 5.1) is used to create a file of EVENT names ordered alphabetically. Then the Formatted Problem Statement Report can be produced. The result is a listing of the test cases/points in the preferred order based on the name of the EVENT.

4.13 Prepare Specification Documentation (BLOCK 12)

Specification documents serve to record the requirements, design, and product descriptions of a system. Specifications are used

throughout the system life cycle and are an integral part of the system management concept: contracting and development, configuration management, system integration and testing, maintenance, and modifications.

Documentation is developed according to different standards depending on the type of system being defined. Where the system requirements are for a military system which can be satisfied by an automated data processing (ADP) configuration of computer hardware and software, the DoD Standard 7935.1-S, Automated Data Systems Documentation Standards, is usually applied. In military systems where ADP hardware and software may be part of a larger system or equipment, such as a weapons system, MIL-STD-490, Military Standard Specification Practices, may be the required documention standard. Although the formats and content requirements of these two standards vary, each can draw upon the products of the analysis performed in the preceding BLOCKS.

The system requirements definition and analysis activities (BLOCKS 3-11) provide the basis upon which the preparation of specification shall proceed. The products of BLOCKS 3-11 (functional and informational hierarchical structures, information and control flows, etc.) shall be incorporated directly into the specification documents in accordance with the prescribed format of the documentation standard by using the analyzer reports as figures or appendices in the specification documents. Additional specification document paragraph headers and text may be required to complete the document in order to explain the analyzer reports or provide continuity between the reports and the format requirements of the standard. Where additional text is required to satisfy the documentation standard format, the added text shall be direct and succinct. The text shall be free of vague and ambiguous terms. The text shall use the simplest words and phrases which convey the intended meaning. Care shall be taken to ensure that the supplemental text does not conflict with previously defined system requirements [BLOCKS 3-11]. Where conflicts arise, the previous requirements definitions and analysis shall take precedence, any conflicts in the supplemental text shall be removed.

The intent of the text is to provide supplemental understanding of the requirements identified and analyzed previously. The style of writing shall emphasize short and concise sentence structure. Well-written sentences shall be required with a minimum of punctuation. Punctuation shall be used to aid reading and prevent misunderstandings. When extensive punctuation is required for clarity, the sentence shall be restructured to eliminate the deficiency. The emphasis shall be upon short and concise sentences and the elimination of compound clauses. Additional style, format and general instructions for preparation of specification documents shall be accomplished as described in DoD Standard 7935.1-S, Part 2 , and MIL-STD-490, paragraph 3.2.

Only a few specification types in DoD Standard 7935.1-S and MIL-STD-490 are used to record the results of the system modeling addressed in this guidebook. These are as follows:

DoD Standard 7935.1-S

Functional Description (FD)
Data Requirements Document (RD)
System/Subsystem Specification (SS)

MIL-STD-490

System Specification (Type A)
Development Specifications (Type B)

The formats of each of these specification types are uniquely different and do not easily lend themselves to the outputs of the analyzer. As stated previously, the analyzer reports can be used as figures or as appendices to support the paragraph requirements of the documentation standards. Where these documentation standards are not required, the analyzer outputs identified in the previous activities [BLOCKS 3-11] can be used as specifications for the target system. However, additional supporting text to explain the analyzer reports and provide a comprehensive specification of the system will be required. The Print Requirements report (3.2X) has been developed to provide automated documentation directly from the requirements data base as described in Appendix C and as illustrated in Appendix D.

4.14 Perform Traceability Analysis [BLOCK 13]

Traceability gives the analyst a means of verifying the requirements by linking each requirement to the varying forms of source documentation such as program directives and plans, studies, analyses, test plans, associated specifications (FD, RD, SS or Type A and B) and the like. Throughout the requirements engineering activities the need exists for the analyst to be able to evaluate the impact of changes and additions to the requirements (Figures 10 and 11). Whatever the reason (policy, economics, study or analysis results, engineering change proposals, etc.), traceability provides the capability to readily identify associated impacts to the system definition as well as to trace the impacts to all other associated documentation. Requirement change impacts can be readily analyzed and the appropriate actions taken where the sources and traces of requirements are identified. The links to associated plans, analyses, studies, and specifications accomplished prior to, during, and subsequent to the start of formal requirements engineering are crucial to the integrity of the system being

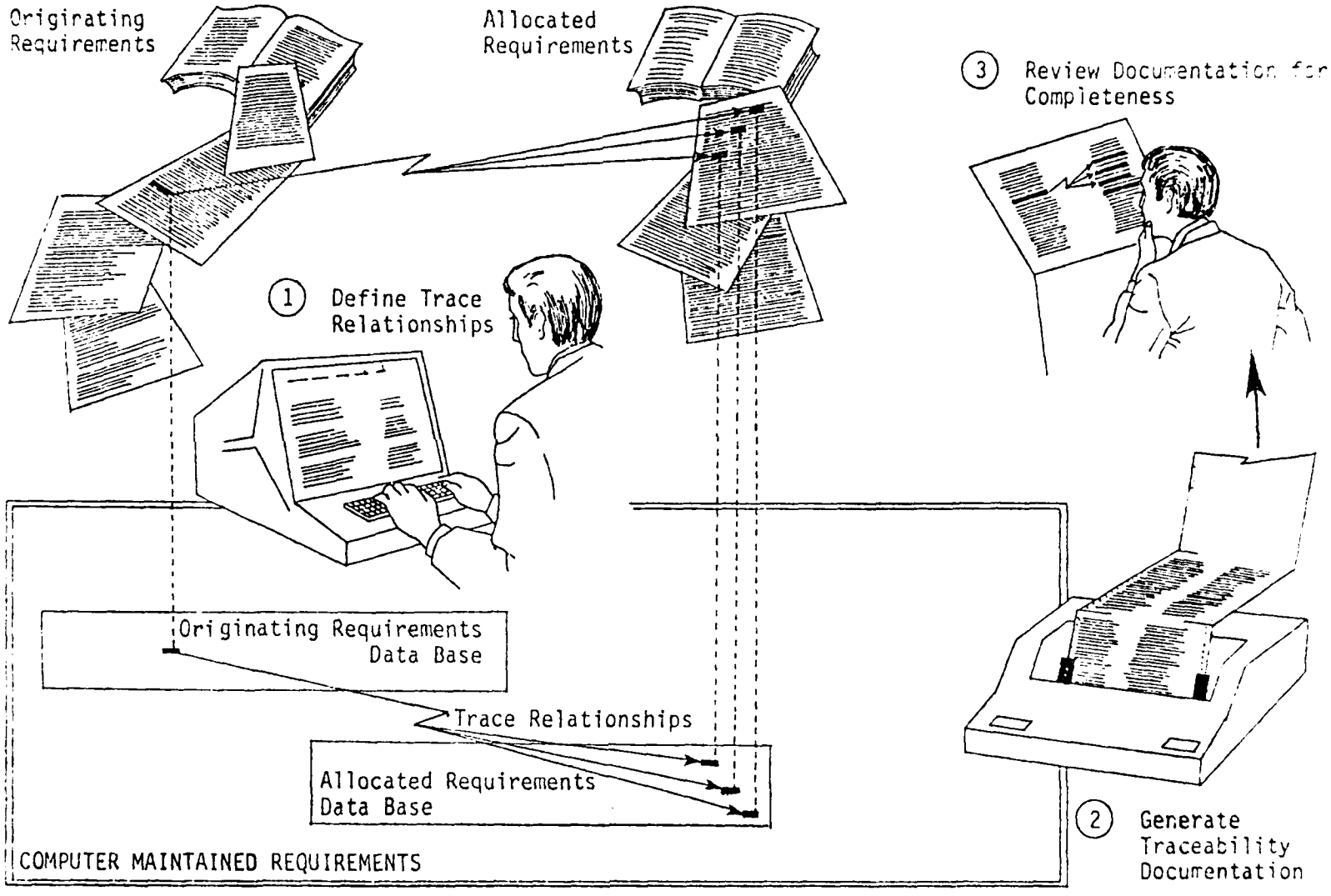


Figure 10. Requirements Traceability Analysis

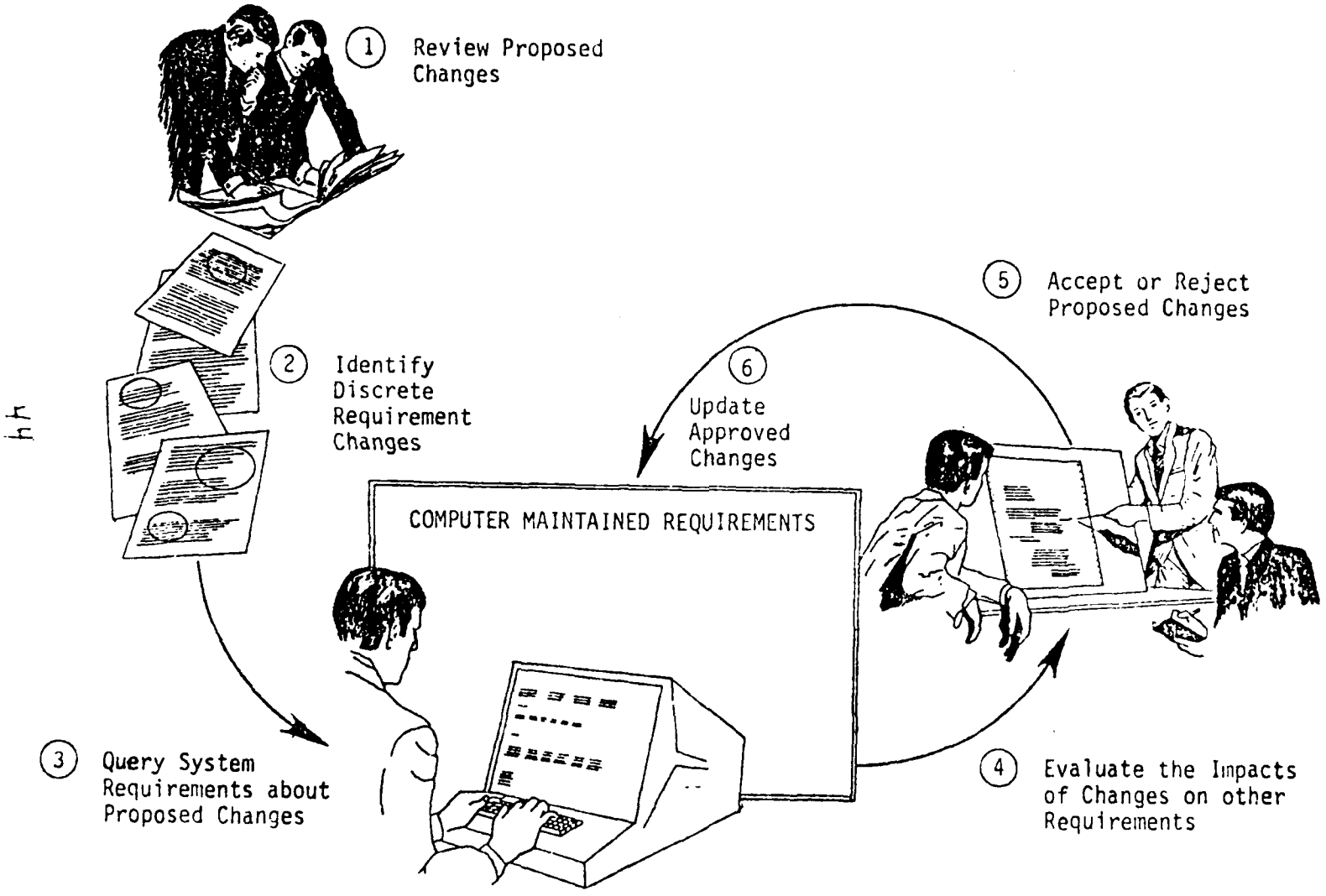


Figure 11. Requirements Configuration Control

defined and developed.

There are two forms of traceability: traces from the originating requirements to the logical model and traces from the logical model to other models where the requirements have been allocated such as more detailed logical models, physical designs (design specifications), and as-built products (product specifications). The SOURCES and TRACE-KEY statements are employed to address both forms of traceability.

SOURCE statements are used during the logical modeling as described in BLOCKS 3-11 to identify the originating source of the requirement (objects: PROCESS, INTERFACE, SET, INPUT, OUTPUT, ENTITY, GROUP/ELEMENT etc.). This can be done by use of a unique SOURCE name (source-identifier). This identifier is often the page and/or paragraph of a source document. The more specific the identifier, the easier it is to locate the source of the requirement, especially by individuals who were not involved in the analysis. Where multiple source documents are to be referenced, the source-identifier should begin with a prefix (usually one character is sufficient) to distinguish the unique source. For example, J-3.1.1.2 represents paragraph 3.1.1.2 in a document identified by the project analysts as "J".

Traces to allocated requirements are accomplished using the TRACE-KEY statement. The preferred method is to add TRACE-KEY statements to the PSL/PSA data base of the logical model where the requirement is first recorded (i.e., the originating model). For instance, TRACE-KEY statements would be added to the logical model (originating model) when the requirements have been allocated or refined (expanded upon) in another more detail model (allocated model). The name of the TRACE-KEY (trace-identifier), like the source-identifier for SOURCES, can be to the page and/or paragraph of the document where the requirement was allocated or the actual name of the allocated requirement (object) in the second model as maintained in a PSL/PSA data base.

Throughout the requirements engineering activities (BLOCKS 3-11) each requirement shall be associated with the sources of the requirement (source documents). These SOURCES shall relate the system requirements to all associated specifications, studies, analyses, plans, program management directives and plans, system sizing and timing studies, prototyping, simulations, test planning, and the like. SOURCE statements shall be included for each requirement type (language objects) as appropriate to the analysis. The source-identifiers should be specific enough that the requirement can be located on a single page of the source document. TRACE-KEYS shall be added to the originating PSL/PSA data base when the requirements in the originating model have been allocated to more detailed models; this shall be accomplished even if the allocated model is not actually defined in a PSL/PSA data base. The name of the TRACE-KEY should be either the object name in the allocated model (PROCESS,

INTERFACE, SET, INPUT, OUTPUT, ENTITY, GROUP/ELEMENT etc.) or the location of the requirement in the allocated model (i.e., document page/paragraph numbers). Since the allocated model may not be represented in a PSL/PSA data base, the page/paragraph numbers of the requirements in the allocated document are often the only appropriate TRACE-KEYS that can be added to the originating model. The following example shows the identification of SOURCES and TRACE-KEYS for a PROCESS:

```
PROCESS validate-time-cards;
SOURCE  j-3.1.1.2, j-3.1.1.2.1;
TRACE-KEY  n-3.7.1.2, n-3.7.1.2.1.3, t-4.3.5.3;
```

In this example two unique references (SOURCES) have been identified for the originating requirement and three references (TRACE-KEYS) have been identified as the location of the requirements in two other documents (n- and t-). The SOURCE statements indicate that the function (PROCESS) "validate-time-cards" originated in document 'j' and is discussed in paragraphs 3.1.1.2 and 3.1.1.2.1. The function traces to requirements allocated and refined in two other documents: document 'n' paragraphs 3.7.1.2 and 3.2.1.2.1.3 and document 't' paragraph 4.3.5.3.

Only a few reports display SOURCES and TRACE-KEYS. These are the Formatted Problem Statement Report, Requirements Traceability Analyzer Report (3.2X) and the Structure Report (3.2X). The Requirements Traceability Analyzer Report has been specifically designed to analyze the traceability of requirements and display the results of the trace between two PSL/PSA data bases in a single report.

4.15 Perform Consistency and Completeness Analysis (BLOCK 14)

Throughout the requirements engineering activities (BLOCKS 3-13) analysis of the consistency and completeness of the requirements definition assures the integrity of the system being defined. The analyzer reports assist the requirements engineer in consistency and completeness analysis by (1) enforcing consistency and unambiguity by checking the syntax of the requirements statements (2) detecting some types of logic errors in the requirements statements and (3) aiding the detection of incomplete and inconsistent requirements statements. By far the majority of inconsistencies and incompletenesses are detected by the requirements engineer as opposed to the analyzer. This is done by observation of the analyzer reports as the engineer becomes highly associated with the problem that is being modeled. Various reports have built-in analysis features which detail certain classes of syntax or logic errors.

In using the reports identified in the previous BLOCKS and other reports which may be employed as described in Appendix C, the following minimum consistency and completeness checks shall be performed.

4.15.1 Identify System Functions: BLOCK 3

Are all functions (PROCESSES) defined in operational terms as opposed to solution oriented terminology such as data processing terms? Remove or rename all functions which imply "how-to".

Are the functions backed by studies or the like which resolve technical risks? Remove all functions which are not feasible or analyze the risks and resolve any uncertainty.

Are all source references (SOURCES) identified for each function?

Have high level functions been broken down into the lowest level functions (functional primitives)? Do all functional primitives have a PROCEDURE defined?

Can any functions be consolidated? Can duplicated or similar functions be eliminated or consolidated?

Have all traces (TRACE-KEYS) been defined for each functional primitive?

4.15.2 Organize Functions into a Hierarchical Structure: BLOCK 4

Does the hierarchical structure contain all functions defined? That is, are all PART/SUBPART relationships entered and correct?

Does the sum of the activities of each set of lower level functions represent the activities of the function at the next higher level in the functional hierarchy? Are there any missing lower level functions?

Does each level of the functional hierarchical structure consist of 2-7 functions? If not, can the hierarchy be restructured?

4.15.3 Identify System Constraints: BLOCK 5

Have all constraints (MEMOS, ATTRIBUTES, HAPPENS) been associated with specific function levels in the functional hierarchy? Are constraint requirements applied to the appropriate functions?

Are the constraints identified by type: pr-, py-, op-, dn- ?

Do constraints have source document references (SOURCES) defined? Each constraint shall be backed by documentation which provides the rationale, or feasibility for the constraint. If no source reference is identified or available, the constraint shall be eliminated.

Do any combinations of constraint requirements imposed on the functions result in excessive or unrealistic engineering requirements, thereby increasing costs, technical and schedule risks during the system development and system life cycle? Where uncertainty or conflicts exist, further analysis shall be performed. As a result the conflicts shall be removed by eliminating or adjusting the conflicting requirements.

Is each constraint requirement defined in quantifiable terms: single values or range of values, including units of measure, limits, accuracy or precision, and frequency?

Have constraints been overspecified? Excessive constraints eliminate design flexibility.

Have all traces (TRACE-KEYS) been defined for each constraint?

4.15.4 Identify System Using Activities BLOCK 6

Have all using activities (organizations, operational units, or positions) been identified and related to associated inputs and outputs?

Have all using activity source references (SOURCES) and traces (TRACE-KEYS) been identified?

4.15.5 Identify External System Inputs-Outputs: BLOCK 7

Have all external system I/O been identified?

Have all required external I/O formats (messages, etc.) been identified and described?

Are all external I/O associated with using activities (BLOCK 6) and functions (BLOCK 6)?

Are all external I/O source document references (SOURCES) and traces (TRACE-KEYS) identified?

4.15.6 Perform Information-Flow Analysis: BLOCK 8

Is there an information-flow sequence defined for every external output desired? Can every external output be traced to inputs?

Is each information-flow sequence complete and logically correct? The information flow shall indicate only the relationship between system functions and system information (external and internal system I/O) and shall not imply any lapse in time or intermediate I/O being used, derived, or updated.

Are all information-flow relationships (USES, DERIVES, UPDATES, GENERATES, RECEIVES) described as appropriate in each information-flow sequence?

Are all using activities (BLOCK 6) associated with system external I/O?

Is each information-flow sequence referenced to source documentation (SOURCE) which established the need for the information-flow sequence as well as resolves any uncertainty or technical risks?

4.15.7 Structure System Information: BLOCK 9

Does the information hierarchy structure contain all external and internal I/O as described in the source documentation (SOURCES)?

Does the sum of the I/O at a given level represent the total contents of the I/O at the next higher level in the hierarchy?

Do the I/O structures represent the contents of each external I/O (SET, INPUT, OUTPUT, GROUP/ELEMENT)? Each internal I/O (SET, ENTITY, GROUP/ELEMENT)?

Are traces (TRACE-KEYS) complete for all I/O?

4.15.8 Perform Control-Flow Analysis: BLOCK 10

Is each control-flow sequence complete and logically correct? No lapse in time or intermediate activity shall be implied between functions in the control-flow sequence.

Are all conditions which determine the flow direction described using the control-flow relationships: TRIGGERS, UTILIZES, CONDITION?

Are initial control flows primarily functional flows? That is, are TRIGGERS and UTILIZES relationships primarily used?

Is each control-flow sequence referenced to source documentation (SOURCES) which establishes the need and rationale for the control-flow sequence as well as resolves any uncertainty of technical risks?

4.15.9 Perform Test Analysis: BLOCK 11

Are all test points (EVENTS) identified?

Are source references (SOURCES) to Test Cases, Test Plans and Procedures, Quality Assurance Provisions etc. identified for each test case or point?

Are test points associated with the control flows? That is, is every test point related to at least one PROCESS in the control flow using CAUSED BY/TRIGGERS statements?

4.15.10 Prepare Specification Documentation: BLOCK 12

Have all requirements defined during BLOCKS 3-11 been incorporated into the appropriate specification paragraphs as figures or appendices without change?

Has supplemental text been restricted and concisely written as described in BLOCK 12?

Has supplemental text been reviewed to identify any conflicts between the text and the system requirements defined in BLOCKS 3-11? Remove any conflicts in the text or reaccomplish the analysis to resolve deficiencies.

4.15.11 Perform Traceability Analysis: BLOCK 13

Have SOURCES been defined for all system requirements as specified in the Requirements Engineering Plan (BLOCK 2)?

Have all system requirements which have no source references been eliminated? If the requirement has no sources, it is not a user requirement.

Have TRACE-KEYS been defined which show the allocated requirements in other models or specifications as required by the Requirements Engineering Plan (BLOCK 2)?

4.16 Manage Requirements Engineering Activities (BLOCK 15)

The preceding BLOCKS describe the activities to be performed in developing a logical model of system requirements. During the modeling activities, project and technical managers often need information which describes (1) the status of the modeling activities from week to week, (2) the quality of the requirements definition as maintained in the requirements data base, and (3) the size and growth of the requirements data base. Most analyzer reports described in the preceding BLOCKS serve the requirements engineers in determining the consistency and completeness of the definition and can be used to document the system. There are, however, several reports which are more specifically intended for project and technical management of the requirements engineering activities.

Reports which aid monitoring the progress being made are the Attribute Report, Data Base Summary Report, and Data Base Status Report. Each of these reports provides statistics on the number of objects and relationships between objects in the requirements data base. User options in these reports allow a variety of displays (counts and/or percentages) which can be used from week to week or over longer periods of time to track various aspects of the requirements data base. For instance, the following status ATTRIBUTES can be used by the requirements engineer to make a statement about the quality of the requirements (PROCESS, INPUT, OUTPUT, etc.) in the requirements data base:

Attribute-Name	Attribute-Value	Meaning
status	ambig	ambigious
status	compl	complete
status	incpl	incomplete
status	incst	inconsistent
status	redun	redundant

These ATTRIBUTES can be associated with any requirement (object). The following example is similar to the performance ATTRIBUTE example shown in [BLOCK 5]:

```
PROCESS  validate-time-cards;
ATTRIBUTES ARE  status  incomplete;
```

In the above example, the ATTRIBUTE is not further described using the DEFINE section as was done in BLOCK 5, because the naming convention described above for the Attribute-Names and Attribute-Values is sufficient for status monitoring purposes.

The Attribute Report and Data Base Status report display these attributes and attribute-values. The project or technical manager can therefore see the quality (status) of the requirements shift from one of poor quality (ambigious, incomplete, inconsistent, or redundant), as might be the case in the initial stages of the analysis, to one of high quality (complete) as the requirements engineering activities are finished. During the requirements engineering project, the requirements data base will gradually approach "complete status" as attributes are changed by the analysis team. The status attributes not only report the progress being made, but also the quality of the requirements themselves (ambigious, inconsistent, redundant) as determined by the analyst. As the counts of each status attribute change over previous weeks, the relative growth of the data base becomes apparent. There are no analyzer reports which display the actual physical size of the requirements data base on the computer where it is hosted or provide information on how much storage remains.

Some reports aid in the configuration management of the requirements data base by maintaining a history of changes made. These are List Changes, the Name-List and Formatted Problem Statement Reports (with appropriate options in effect), and Data Base Status. These reports and those previously mentioned are described in Appendix C and further described in the Analyzer references in Appendix A.

APPENDIX A

SELECTED REFERENCES

[1] Automated Data Systems Documentation Standards, Standard 7935.1-S, Office of the Assistant Secretary of Defense, 13 September 1977.

[2] Military Standards Specifications Practices, MIL-STD-490, Department of Defense, October 1968.

[3] D.G. Smith, P.B. Merrithew, Requirements Standards Study (RSS), Volumes I-III, Rome Air Development Center (RADC/ISIE) Contract No. F30602-77-C-0207, Logicon Inc, October 1979.

[4] SAMSO CADSAT Analysis, SAMSO Contract No. F04701-77C-0069 (P00006), Logicon Inc., 28 September 1979.

[5] Program Documentation for Logicon Modifications and Extensions to Computer-Aided Design, Specification, and Analysis Tool (CADSAT), Volumes I-III, USAF/AFSC (ESD/OCT) Contract No. F19628-76-C-0218, 15 April 1978.

[6] User Requirements Analyzer (URA), User's Manual, H6180/Multics/Version 3.2 & Figures. ESD-TR-78-128, ISDOS Project, University of Michigan, USAF/AFSC: Electronic Systems Division (ESD/TOI), Hanscom AFB, MA, March 77.

[7] User Requirements Language (URL) User's Manual Part I & II, H6180/Multics/Version 3.2, ESD-TR-78-127 & ESD-TR-78-129, ISDOS Project, University of Michigan, USAF/AFSC: Electronic Systems Division (ESD/TOI), Hanscom AFB, MA, March 77.

[8] Problem Statement Language (PSL) Introduction and User's Manual, PSA Version 4.2, ISDOS Project, University of Michigan, ISDOS Ref. No. 7742-0143-0, May 1977.

[9] Problem Statement Analyzer (PSA) Reports and Report Commands, Version A4.2, ISDOS Project, University of Michigan, ISDOS Ref. No. 7742-0144-1, December 1977.

[10] Problem Statement Analyzer (PSA) Modifier Commands, Version A4.2, ISDOS Project, University of Michigan, ISDOS Ref. No. 7742-0145-1, December 1977.

[11] Problem Statement Language (PSL) Language Reference Summary, Version A5.1, ISDOS Project, University of Michigan, ISDOS Ref. No. 79A51-0174-4, September 1979.

[12] Problem Statement Analyzer (PSA) Reports and Report Commands, Version A5.1, ISDOS Project, University of Michigan, ISDOS Ref. No. 79A51-0173-5, August 1979.

[13] Problem Statement Analyzer (PSA) Modifier Commands, Version A5.1, ISDOS Project, University of Michigan, ISDOS Ref. No. 79A51-0178-5.

[14] D. Ross, "Structured Analysis (SA): A Language for Communicating Ideas," IEEE Transactions on Software Engineering, Vol. 3, No. 1, January 1977.

[15] T. DeMarco, Structured Analysis and System Specification, New York: Yourdon, Inc., 1978.

[16] G. J. Myers, Reliable Software Through Composite Design, Petrocelli/Charter, New York, 1975.

[17] E. Yourdon and L. Constantine, Structured Design, New York: Yourdon, Inc., 1975.

[18] M. Jackson, Principles of Program Design, New York, NY, Academic Press, 1975.

[19] C. Gane and T. Sarson, Structured Systems Analysis: Tools and Techniques, Prentice-Hall, Englewood Cliffs, N.J., 1979.

[20] Tutorial on Software Design Techniques, second edition, IEEE Computer Society, 1977.

APPENDIX B

SELECTED LANGUAGE FEATURES

This appendix provides a condensed list of the language features which support the requirements engineering activities (BLOCKS) described in this guidebook. In general, the language features presented here will provide the requirements engineer with adequate capabilities which will make the requirements definition, analysis, and documentation process productive and meet the objectives of the activities expressed in this guidebook. This appendix provides a quick-reference list for beginning users of the language and aids in determining the language features to be employed. It is expected that in a given application this set of language objects and statements within the object sections will be either reduced or expanded to meet the objectives of a specific application. This list will provide a basis for the selection process. Further details concerning the language features are described in the language reference documents available for the tool version being used (Appendix A). Availability of any language feature which has changed or has been added from one version to another is noted. In addition, a cross reference is provided between the language feature and the BLOCKS to which they apply.

The versions are denoted by the following numbers used in this appendix:

(3.2) URL/URA (CARA or CADSAT), an Air Force version of PSL/PSA, University of Michigan (ISDOS Project), 1974.

(3.2X) 3.2 plus extensions and modifications made by Logicon Inc. for the Air Force, 1976.

(4.2) PSL/PSA version available from the University of Michigan (ISDOS Project), 1977-1978.

(5.1) Most recent version of PSL/PSA available from the University of Michigan (ISDOS Project), 1978-1979.

The following sections (objects) are contained in this appendix:

SECTIONS	CROSS REFERENCE
ATTRIBUTE	BLOCKS 5 and 15
CONDITION	BLOCK 10
ELEMENT	BLOCKS 7 - 9
ENTITY	BLOCKS 8 - 9
EVENT	BLOCK 11
GROUP	BLOCKS 7 - 9
INPUT	BLOCKS 7 - 9
INTERFACE	BLOCK 6
MEMO	BLOCK 5
OUTPUT	BLOCKS 7 - 9
PROCESS	BLOCK 3
SET	BLOCKS 7 - 9

1.0 SELECTED LANGUAGE FEATURES COMMON TO ALL SECTIONS (OBJECTS)

```
ATTRIBUTES ARE attribute-name-1  attribute-value-1, [BLOCKS 5,15]
 (attr)                 .                 .
                        .                 .
                        .                 .
               attribute-name-n  attribute-value-n;      (1)

DESCRIPTION (desc);

-----------------------------                            (2)
-----comment-entry------------
-----------------------------;


SYNONYM (syn)  synonym-name(s);                          (3)


KEYWORD (key)  keyword-name(s);


SOURCE (src)  source-identifier(s);


TRACE-KEY (tkey) trace-identifier(s);


SEE-MEMO (sm) memo-name(s);
```

NOTES:

(1) ATTRIBUTES for an object are initially identified (ATTRIBUTE-NAME and ATTRIBUTE-VALUE only) within the section (object) to which they apply by use of the ATTRIBUTE (attr) statement. The ATTRIBUTE section (2.0) provides the means of elaborating on the ATTRIBUTE through use of the statements such as desc, syn, key, src, tkey, and sm.

(2) Text descriptions (comment-entries) can be used to expand upon the object where the syntax of the language does not suffice. Comment-entries are free format, 72 characters per line, maximum of 60 lines. Comment-entries should be limited in most applications since the data base grows considerably with their use. Succinct comment-entries are advised.

(3) Where requirement names (objects) in this appendix end in "(s);", more than one object may be identified. Each object is separated by a comma and the last object ends with a colon.

2. ATTRIBUTE SECTION [BLOCKS 5 and 15]

ATTRIBUTES are used to define other aspects about an object which cannot be done by other statements provided in the language. ATTRIBUTES for an object are initially identified (ATTRIBUTE-NAME and ATTRIBUTE-VALUE only) within the section (object) to which they apply by the use of the ATTRIBUTE (attr) statements as indicated in 1.C. This section provides the means of elaborating on the previously identified ATTRIBUTE through use of any of the statements within the ATTRIBUTE section (i.e., desc, syn, key, src, tkey, sm);

There are two ways ATTRIBUTES are used in this guidebook: (1) as one means of defining constraints [BLOCK 5] and (2) to define status attributes [BLOCK 15]. Constraint and status attributes can be defined by using the following naming conventions:

Constraint Attributes: [BLOCK 5]

Attribute-Name	Attribute-Value
pf-attribute-name	performance-constraint
py-attribute-name	physical-constraint
op-attribute-name	operability-constraint
dn-attribute-name	design-constraint

Status Attributes: [BLOCK 15]

Attribute-Name	Attribute-Value	Meaning
status	ambig	ambigious
status	compl	complete
status	incpl	incomplete
status	incst	inconsistent
status	redun	redundant

DEFINE (def) attribute-name ATTRIBUTE; (1)

desc, syn, key, src, tkey, sm
See Section 1

NOTE

(1) Versions 3.2 and 4.2 have a DEFINE section for describing certain objects in greater detail. The ATTRIBUTE is one object in 3.2 and 4.2 that can be expanded upon using the DEFINE section as shown above. In version 5.1 there is no DEFINE section and the ATTRIBUTE has become a new section. Version 5.1 requires the DEFINE (def) to precede each section. Thus "PROCESS process-name;" becomes "DEFINE PROCESS process-name;". This appendix reflects the format for each language section which is compatabile with versions 3.2 and 4.2; users of version 5.1 will have to add "DEFINE" preceding each section header statement.

3. CONDITION SECTION (BLOCK 10)

This section provides the means to define the conditions which control the direction of system control flow. The condition may be initially identified (condition-name only) under the PROCESS section by the use of the TRIGGERED WHEN relationship. The condition section provides a means of elaborating on the condition. In addition, this section allows a previously unidentified condition which was omitted in the associated process definition to be identified (named), associated with functions (PROCESSES) and described.

CONDITION (cond) condition-name (BLOCK 10)

attr, desc, syn, key, src, tkey, sm
See Section 1

BECOMING TRUE TRIGGERS (becg t trgs) process-name(s);
BECOMING FALSE TRIGGERS (becg f trgs) process-name(s);

TRUE WHILE (t whl);

-----comment-entry----------;

FALSE WHILE (f whl);

-----comment-entry----------;

4. ELEMENT SECTION (BLOCKS 7-9)

```
ELEMENT (ele) system-information-name;                    (BLOCKS 7-9)

 ---------------------------------------
|                                       |
| attr, desc, syn, key, src, tkey, sm   |
|                                       |
|           See Section 1               |
 ---------------------------------------

CONTAINED IN (cntd) {  group-}                              (BLOCK 9)
                    { entity-}
                    {  input-}
                    { output-}name(s);

DERIVED BY (drvd)  process-name(s);    (1)                  (BLOCK 8)
UPDATED BY (updd)  process-name(s);    (1)                  (BLOCK 6)
USED BY (used)     process-name(s);    (1)(2)               (BLOCK 6)

VALUES ARE (val) value-name(s) [THRU value-name(s)];        (BLOCK 8)
```

NOTE

(1) See note (5) under the PROCESS section
(2) EMPLOYED BY (epld) in version 5.1

5. ENTITY SECTION (BLOCKS 8-9)

```
ENTITY (ent) internal-information-name;                    (BLOCKS 8-9)

 ------------------------------------
|                                    |
| attr, desc, syn, key, src, tkey, sm |
|                                    |
|          see Section 1             |
 ------------------------------------

PART OF (part) entity-name;                                   (BLOCK 9)
SUBPARTS ARE (subp) entity-name(s);                           (BLOCK 9)

CONSISTS OF (csts) { group-}                                  (BLOCK 9)
                   {element-}name(s);

CONTAINED IN (cntd) set-name(s);          (1)                 (BLOCK 9)

DERIVED BY (drvd)  process-name(s);       (2)                 (BLOCK 8)
UPDATED BY (updd)  process-name(s);       (2)                 (BLOCK 8)
USED BY    (used)  process-name(s);       (2)(3)              (BLOCK 8)
```

NOTES

(1) COLLECTED IN (cltd) in version 5.1
(2) See note (5) under the PROCESS section
(3) EMPLOYED BY (epld) in version 5.1

6. EVENT SECTION [BLOCK 11]

```
EVENT (ev) event-name;                                    [BLOCK 11]

 ------------------------------------
|                                    |
| attr, desc, syn, key, src, tkey, sm |
|                                    |
|          See Section 1             |
 ------------------------------------

PART OF (part) event-names;                               [BLOCK 11]
SUBPARTS ARE (supp) event-name(s);                        [BLOCK 11]

CAUSED BY (csd)  process-name(s);                         [BLOCK 11]
TRIGGERS (trgs)  process-name(s);                         [BLOCK 11]
```

7. GROUP SECTION (BLOCKS 7-9)

```
GROUP (gr) system-information-name;                          (BLOCKS 7-9)

 ------------------------------------
|                                    |
| attr, desc, syn, key, src, tkey, sm |
|                                    |
|          See Section 1             |
 ------------------------------------

CONSISTS OF (csts)  {  group-}                                (BLOCK 9)
                    {element-}name(s);

CONTAINED IN (cntd) {entity-}                                 (BLOCK 9)
                    { input-}
                    {output-}
                    { group-}name(s);

DERIVED BY (drvd)  process-name(s);      (1)                  (BLOCK 8)
UPDATED BY (updd)  process-name(s);      (1)                  (BLOCK 8)
USED BY (used)     process-name(s);      (1)(2)               (BLOCK 8)
```

NOTES

(1) See notes (5) under the PROCESS section
(2) EMPLOYED BY (epld) in version 5.1

8. INPUT SECTION (BLOCK 7-9)

```
INPUT (inp)  external-information-name;                    [BLOCK 7]

 ------------------------------------
|                                    |
| attr, desc, syn, key, src, tkey, sm |
|                                    |
|          See Section 1             |
 ------------------------------------

CONSISTS OF (csts)  { group-}                              [BLOCK 9]
                    {element-}name(s);

CONTAINED IN (cntd)  set-name(s)          (1)              [BLOCK 9]

PART OF (part) input-name;                                 [BLOCK 9]
SUBPARTS ARE (supp) input-name(s);                         [BLOCK 9]


GENERATED BY (gend)  interface-name(s);                    [BLOCK 8]
RECEIVED BY (rcvd)   process-name(s);                      [BLOCK 8]
USED BY (used)       process-name(s);      (2)(3)          [BLOCK 8]
```

NOTES

(1) COLLECTED IN (cltd) in version 5.1
(2) EMPLOYED BY (epld) in version 5.1
(3) See note (5) under the PROCESS section

9. INTERFACE SECTION (BLOCK 6)

```
INTERFACE (intf)  using-activity-name(s);          (1)     (BLOCK 6)

 ------------------------------------
|                                    |
| attr, desc, syn, key, src, tkey, sm |
|                                    |
|           See Section 1            |
 ------------------------------------

PART OF (part) using-activity-name;                        (BLOCK 6)
SUBPARTS ARE (subp)  using-activity-name(s);               (BLOCK 6)

GENERATES (gens)  input-name(s);                     (BLOCK 6, 8)
RECEIVES (rcvs)  output-name(s);                     (BLOCK 6, 8)
```

NOTE

(1) The PROCESS can also be used to represent using activities in lieu of INTERFACES, see BLOCK 6.

10. MEMO SECTION [BLOCK 5]

The MEMO section can be used in two ways: (1) as a means of recording analysts notes (Note Memo) , and (2) as one means of describing system constraints: performance, physical, operability, and design requirements (Constraint Memo), see BLOCK 5. The name of the constraint memo includes the prefixes as indicated in note 1.

MEMO (memo) prefix-memo-name; (1) [BLOCK 5]

attr, desc, syn, key, tkey, src
see Section 1

APPLIES TO (app) non-memo-name; (2) [BLOCK 5]

NOTES

(1) The prefix is used to distinguish the various constraints. When the MEMO is being used to define a constraint the prefixes identified below shall be applied. For note memos the prefix is omitted.

pf-	performance
py-	physical
op-	operability
dn-	design

(2) Where the MEMO is used to define a constraint, the name of the memo (non-memo-name) shall be the process-name. This identifies the function that is being constrained.

100. OUTPUT SECTION (BLOCKS 7-9)

```
OUTPUT (out) external-information-name;                    [BLOCK 7]

 ------------------------------------
|                                    |
| attr, desc, syn, key, src, tkey, sm |
|                                    |
|          See Section 1             |
 ------------------------------------

CONSISTS OF (csts)  { group-}                              [BLOCK 9]
                    {element-}name(s);

CONTAINED IN (cntd)  set-name(s);          (1)             [BLOCK 9]

PART OF (part) output-name;                                [BLOCK 9]
SUBPARTS ARE (supp) output-name(s);                        [BLOCK 9]


GENERATED BY (gend) process-name(s);                       [BLOCK 8]
DERIVED BY (drvd)   process-name(s);       (2)             [BLOCK 8]
RECEIVED BY (rcvd)  interface-name(s);                     [BLOCK 8]
```

NOTES

(1) COLLECTED IN (cltd) in version 5.1
(2) See note (5) under the PROCESS section

11. PROCESS SECTION [BLOCK 3]

The PROCESS section is used to describe the target system functions. The following is a summary of selected PROCESS language features which are applicable to the requirements engineering activities [BLOCKS] described in this guidebook.

```
PROCESS (prc)  process-name;                          (1)    [BLOCK 3]

 ------------------------------------
|                                    |
| attr, desc, syn, key, tkey, src, sm |
|                                    |
|          see Section 1             |
 ------------------------------------

PART OF (part)   process-name;                                [BLOCK 4]
SUBPARTS ARE (subp)  process-name(s);                         [BLOCK 4]

TRIGGERS (trgs)  process-name(s)                             [BLOCK 10]
TRIGGERED BY (trgd)  process-name(s);                        [BLOCK 10]
TRIGGERED WHEN  condition-name  BECOMES {TRUE      (2)      [BLOCK 10]
                                        {FALSE;

UTILIZES (utls)  process-name(s);                            [BLOCK 10]
UTILIZED BY (utld)  process-name(s);                         [BLOCK 10]

             {    set-}                          (3)(5)      [BLOCK 8]
             {  input-}
USES (uses)  { entity-}name(s);
             {  group-}
             {element-}
                                                             [BLOCK 8]
                                    (4)(5)
             {    set-}        [              {    set-}        ]
             {  input-}        [    {DERIVE}  { output-}        ]
USES (uses)  { entity-}name(s) [ TO {       }  { entity-}name(s) ];
             {  group-}        [    {UPDATE}  {  group-}        ]
             {element-}        [              {element-}        ]
```

```
                                                                  [BLOCK 8]
                                                  (5)
              {    set-}          [              {    set-}          ]
              { output-}          [              {  input-}          ]
DERIVES       { entity-}name(s)   [     USING    { entity-}name(s)   ];
 (drvs)       {  group-}          [              {  group-}          ]
              {element-}          [              {element-}          ]


                                                                  [BLOCK 8]
                                                  (5)
              {    set-}          [              {    set-}          ]
              { entity-}          [              {  input-}          ]
UPDATES       {  group-}name(s)   [     USING    { entity-}name(s)   ];
 (upds)       {element-}          [              {  group-}          ]
                                  [              {element-}          ]

GENERATES (gens)  output-name(s);                                 [BLOCK 8]
RECEIVES (rcvs)   input-name(s);                                  [BLOCK 8]

PROCEDURE;

-----comment-entry------------;          (6)                      [BLOCK 3]
```

NOTES:

(1) Throughout this appendix the process-name is the actual name of the function identified during BLOCK 3.

(2) See Condition Section, 3.0.

(3) EMPLOYS (epls) in Version 5.1. The compound forms, USES-TO DERIVE and USES-TO UPDATE are better as described in note (5) below.

(4) This form is complementary to the DERIVES-USING and UPDATES-USING statements below.

(5) The PSA information flow reports (Picture, Extended Picture) are more meaningful when using the compound PSL statements verses the simplex forms of just USES, DERIVES, and UPDATES. In addition, compound forms can be expressed under the SET, INPUT, OUTPUT, ENTITY, GROUP, and ELEMENT sections. However, it is recommended that the compound uses be restricted only to the PROCESS section. Therefore, compound forms have been presented only in this PROCESS section.

(6) The PROCEDURE statement is used to describe: (A) the

sequence of operations needed to implement the function (PROCESS), e.g. Structured English, (b) Decision Tables, (c) Decision Trees. PROCEDURE statements should be defined for functional primitives only, i.e, the functions at the lowest level in the functional hierarchy. The use of comment-entries to define a PROCEDURE should be limited just as with DESCRIPTIONS, see note 2, Section 1.

12. SET SECTION [BLOCKS 7-9]

SET information-name; [BLOCK 7]

attr, desc, syn, key, src, tkey, sm
See Section 1

SUBSET OF (sst) set-name(s); [BLOCK 9]
SUBSETS ARE (ssts) set-names(s); [BLOCK 9]

CONSISTS OF (csts) {entity-} (1) [BLOCK 9]
{ input-}
{output-}name(s);

USED BY (used) process-name(s); (2)(3) [BLOCK 8]
DERIVED BY (drvd) process-name(s); (2) [BLOCK 8]
UPDATED BY (updd) process-name(s); (2) [BLOCK 8]

NOTES

(1) COLLECTIONS OF (cltn) in version 5.1
(2) See note (5) under the PROCESS section
(3) EMPLOYED BY (epld) in version 5.1

APPENDIX C

ABSTRACTS OF ANALYZER REPORTS

This appendix contains a list of PSA reports and a brief description of each. This list represents a composite of reports available from the various versions of the analyzer now in use. Versions are denoted by the following numbers adjacent to the report titles:

(3.2) URL/URA, an Air Force version of PSL/PSA, University of Michigan (ISDOS Project), 1974.

(3.2X) 3.2 plus extensions and modifications made by Logicon Inc. for the Air Force, 1976.

(4.2) PSL/PSA version available from the University of Michigan, 1977-1978.

(5.1) PSL/PSA most recent version available from the University of Michigan, 1978-1979.

Some report names have been changed as new versions were released. Many reports include new capabilities over previous releases. The descriptions for each report provided below encompass the most recent capabilities. Refer to the description of the reports for the version available for detailed capabilities and procedures for generating the reports.

PSA provides the capability to create and modify the requirements data base using various modifier commands as described below. PSA also provides the capability to generate reports which aid the requirements engineering activities (BLOCKS) as described in Section 4 of this guidebook. Where a report supports the requirements engineering activities (BLOCKS) described in this guidebook, applicable BLOCKS are denoted at the end of the report description. In the following paragraphs, upper case words are used to indicate the special PSL/PSA reserved words for objects, relationships, and properties of the target system model. For instance, a function is represented by the PSL statement PROCESS.

1.0 Modifier Commands

Combine (4.2)(5.1)

Allows the requirements engineer to combine the information for two objects in the requirements data base and record the combination as one object. A record of this change is produced in the form of the Combined Names report. [All BLOCKS]

Change-Type (3.2)
Change-Name-Type (4.2)(5.1)

Allows the requirements engineer to change the object type defined in the requirements data base. A record of this change is generated in the form of the Change-Type report (3.2) or Change-Name Type report (4.2, and 5.1). [All BLOCKS]

Delete (3.2)
Delete-Name (4.2)(5.1)

Allows the requirements engineer to delete an object name or list of names from the requirements data base. When a name is deleted all of its relationships to other object names in the data base are also deleted. A record of the change is generated in the form of the Deletion report (3.2) or Delete Name report (4.2 and 5.1). [All BLOCKS]

Delete-Comment-Entry (3.2)(4.2)(5.1)

Allows the requirements engineer to delete from the requirements data base narrative text (comment-entries) associated with an object or list of objects. A record of the change is generated in the form of the Deleted Comment Entries report. [All BLOCKS].

Delete-PSL (3.2)(4.2)(5.1)

Allows the requirements engineer to delete selected language statements in the requirements data base. A record of the change is generated in the form of the Deleted PSL report (3.2, 4.2) or Delete-PSL Source Listing and Cross Reference Reports (5.1). [All BLOCKS]

Input-Layout (5.1)

Allows the requirements engineer to enter LAYOUT comment entries in a format which can be processed by the LAYOUT command. A column number heading is given for use during input of LAYOUT comment entries. The Input Layout report is generated to document the LAYOUT. [BLOCK 7]

Input-PSL (3.2)(4.2)(5.1)

Allows the requirements engineer to add requirements to the requirements data base. A record of the additions is generated in the form of the As-Is Source Listing (3.2) or Input-PSL Source Listing and PSA Cross Reference reports (4.2 and 5.1). [All BLOCKS]

Problem-Name (5.1)

Allows the requirements engineer to enter the name of the problem/project into the requirements data base in order that the headings of the PSA reports will contain the name of the project. The Problem Name report is generated to document the change of the project name. [BLOCK 15]

Punch-Comment-Entry (3.2)(4.2)(5.1)

Allows the requirements engineer to retrieve from the requirements data base only the narrative text (comment-entries) for specified objects. The retrieved comment entries are listed on the Punched Comment Entries report and/or output to a host system file. [All BLOCKS]

Rename (3.2)
Change-Name (4.2)(5.1)

Allows the requirements engineer to change the name of an object in the requirements data base. The Rename report (3.2) or Change-Name report (4.2 and 5.1) is produced as a record of the changes. [All BLOCKS]

Replace-Comment-Entry (3.2)(4.2)(5.1)

Allows the requirements engineer to replace, for a given object name, specific narrative text (comment-entries) associated with the object. The Replace Comment Entries report is produced as a record of the change. [All BLOCKS]

Replace-PSL (5.1)

Allows the requirements engineer the means of replacing PSL statements in the requirements data base. A record of the change is recorded in the Replace-PSL Source Listing and Cross Reference reports. [All BLOCKS]

2.0 Report Commands

Assertion-Consistency (5.1)

LIST FORMAT: Shows assertions made (and about) a given object name and the overall consistency of ATTRIBUTE values. [BLOCK 14]

Attribute Report (3.2)(4.2)(5.1)

TABLE FORMAT: Shows all object names in the data base to which an ATTRIBUTE applies and the associated ATTRIBUTE values for the object names. Aids management, and completeness and consistency checking. [BLOCKS 5, 12, 14, 15]

Contents Report (3.2)(4.2)(5.1)

STRUCTURED-LISTING FORMAT: Shows the hierarchy of the data structure based on CONSISTS(COLLECTION)/CONTAINED statements. Automated consistency checking is optional. Report shows a concise listing of the logical information structures to be handled by the target system. [BLOCKS 9, 12, 14]

Consists Comparison Report (3.2)
Contents Comparison Report (4.2)(5.1)

LIST-MATRIX FORMAT: Used to detect redundant or similar data structures; used to optimize data structures. Based on CONSISTS(COLLECTION)/CONTAINED statements. [BLOCK 14]

Consists Matrix Report (3.2)
Contents Analysis Report (4.2)(5.1)

LIST-MATRIX FORMAT: Based on CONSISTS(COLLECTION)/CONTAINED statement; used to detect incomplete or redundant data structures. [BLOCK 14]

Data Process Report (3.2)
Data Process Interaction Report (4.2)
Data-Activity Interaction (5.1)

MATRIX FORMAT: Shows interaction between activities (PROCESSES/INTERFACES) and data objects (SETS, INPUTS, OUTPUTS, ENTITIES, GROUPS, ELEMENTS); used to detect incompletenesses or inconsistencies in data used, including data derivation. Also can be used in data-flow analysis; used by design engineers to plan the logic of target system using the data-activity dependencies. [BLOCK 8]

Data Base Summary (3.2)(4.2)(5.1)

TABLE FORMAT: Technical and management report providing selected presentation of progress being made (status) on the target system. Compared with previous reports; the changes denote progress. Also shows incomplete (undefined) objects. [BLOCKS 14, 15]

Data Base Status (3.2X)

TABLE FORMAT: Technical and management report providing a listing of requirements (PROCESS and MEMO objects) ordered according to the sources of the requirements; The report includes sources, the PROCESS/MEMO object name, SYNONYM, a cross reference to the structure report, a status attribute value (unambigious, complete, incomplete, inconsistent, redundant, or other user defined attributes) and the number of times the PROCESS/MEMO has been revised. The remaining columns indicate counts showing the status of the PROCESS/MEMO such as the number of synonyms, descriptions, keywords, sources, tracekeys, as well as flow relationships (triggers, utilizes, derives, receives, etc.). This status count is controlled by user option and the display can be changed to satisfy monitoring needs. The report can be compared with previous reports, the changes denote progress. It is also useful in checking the completeness of the analysis of existing source documentation. [BLOCKS 14, 15]

Dictionary Report(3.2)(4.2)(5.1)

NARRATIVE & OUTLINE FORMAT: This report presents selective information on an object (object-name, DESCRIPTION, SYNONYM, KEYWORDS, ATTRIBUTES); a subset of the information presented in the Formatted Problem Statement report. [BLOCKS 12, 14]

Dynamic Interaction (5.1)

MATRIX FORMAT: Shows system sequencing and dynamic states, that is, functional/control flow and events/conditions. Similar information is presented in the Process-Chain report, but this report provides a matrix representation and provides completeness and consistency diagnostics. [BLOCKS 10, 11, 14]

Element Process Analysis Report (4.2)(5.1)

TABLE & MATRIX FORMAT: Aids analysis of interaction between system data and PROCESSES. Designed to accompany the Element Process Usage Report. [BLOCKS 8, 14]

Element Process Usage Report (4.2) (5.1)

TABLE & MATRIX FORMAT: Shows interaction between lowest level data structure objects (usually ELEMENTS) to lowest level PROCESSES; consistency and completeness checking determines that each PROCESSES interacts with some system data. Aids in checking for redundant data structures. May be used in conjunction with the Element Process Analysis Report. [BLOCKS 8, 14]

Extended Picture Report (3.2)(4.2)(5.1)

GRAPHIC FORMAT: A graphical network showing structures for system data or activities (PROCESSES/INTERFACES) and data flow into, within, and out of the target system. [BLOCKS 6, 12, 14]

Formatted Problem Statement(3.2)(4.2)(5.1)

NARRATIVE & OUTLINE FORMAT: Describes an object and its relationships to all other objects, and other descriptive entries about the object. Provides a complete display of all information about a given object (i.e., a complete specification of computer maintained information on the object), formatted in the same manner as it would have been originally entered into the data base. See also Structure Report version 3.2X. [All BLOCKS]

Frequency Report (3.2)(4.2)(5.1)

LISTING FORMAT: Presents system performance (HAPPENS statements) relative to specific INTERVALS. Aids checking all objects related by frequency, understanding the various parts of the system with respect to frequency, and the amount of input/output to be handled by the target system. [BLOCKS 5, 12, 14]

Function Flow Data Diagram (5.1)

GRAPHIC FORMAT: Presents a single activity (PROCESS or INTERFACE) centered on the page with all data objects flowing into the activity on the left and all outputs flowing from the activity on the right. ATTRIBUTES and SYNONYMS are optionally displayed. Slightly different presentation in the Picture Report and Process Summary Report. [BLOCKS 8, 14]

Identifier Information Report (3.2)
Identifier Analysis Report (4.2)(5.1)

MATRIX FORMAT: Shows all information based on the use of IDENTIFIERS for ENTITIES, INPUTS, and OUTPUTS; aids in completeness and consistency checking. [BLOCK 14]

Keyword In Context (KWIC) Report (3.2)(4.2)(5.1)

LISTING FORMAT: Presents logical grouping of object names and permutations of names as maintained in the data base; used for consistency checking or to locate object names when only part of the name is known or assumed. [BLOCK 14]

Layout Report (5.1)

NARRATIVE & OUTLINE: A report showing the layout comment entries. [BLOCK 14]

List Changes (4.2)(5.1)

LIST FORMAT: Shows time and date of each change to the data base and the modifier command used; similar information is available as an output option with Name-List and Formatted-Problem-Statement Reports. Useful in management of the data base. See also Data Base Status report revision count. [BLOCKS 14, 15]

Name-Gen (3.2)
Name-Selection (4.2)(5.1)

LIST FORMAT: This is a report command which generates a file of object names from the data base using a user-defined selection statement. Used to prepare a list of names to be used as input for generation of other reports. See command parameter description (Input-Source) for most reports. [All BLOCKS]

Name List (3.2)(3.2X)(4.2)(5.1)

LIST FORMAT: Lists all names (ordered alphabetically or grouped by object type) in the data base along with the designation of type, synonyms and sources (3.2X only). Useful as a directory and provides alphabetical grouping which is useful in checking on conventions in naming objects such as the use of prefixes in the object name. [All BLOCKS]

Picture Report (3.2)(4.2)(5.1)

GRAPHIC FORMAT: Shows a single object (INTERFACE, PROCESS, SET, INPUT, OUTPUT, GROUP/ELEMENT) and its immediate structure and/or data flow. Useful for high level graphics (snapshots) of system requirements and can be useful in structured walkthroughs as a communications media between analyst and target system user. Used for completeness and consistency checking. See also Function Flow Data Diagram, and Process Summary Report. [BLOCKS 4, 6, 8, 10, 12, 14]

Print Requirements Report (3.2X)

LIST FORMAT: A specification document showing system functions (PROCESSES) and constraints presented in the order of the hierarchical structure of functions. The automated specification document includes narrative text (comment-entries) in association with the functions and constraints which they describe. The report is used as an interface document between the requirements engineer and the target system user and/or to provide intermediate and final documentation (specifications) of the functional requirements of the system. The report is functionally equivalent to DoD Standard 7935.1-S and MIL-STD-490 specification requirements. [BLOCK 12].

Process Chain Report (3.2)(4.2)(5.1)

GRAPHIC FORMAT: Shows system sequencing and dynamic states; that is, functional/control flow and events/conditions. A pictorial network of dynamic relationships between objects such as PROCESSES, EVENTS, CONDITIONS, INPUTS, OUTPUTS. The Dynamic Interaction Report provides similar information in a matrix format. [BLOCKS 10, 11, 12, 14]

Process Input/Output (3.2)
Process Summary (4.2)(5.1)

STRUCTURED-LIST FORMAT: Shows a list of PROCESSES followed by any description of the PROCESS and INPUTS and OUTPUTS of each process. Useful in providing a general description of the target system PROCESSES and associated INPUTS and OUTPUTS. Similar information provided by the Function Flow Data Diagram and Picture Report. [BLOCKS 8, 12, 14]

Punched Comment Entries (3.2)(4.2)(5.1)

LIST, NARRATIVE & OUTLINE FORMAT: Shows narrative descriptions (comment-entries) about an object; aids completeness and consistency checking. [BLOCK 14]

Relation Structure (4.2)(5.1)

TABLE & MATRIX FORMAT: Presents data structure information; used by requirements engineers for completeness and consistency checking of the logical data structure model; used by design engineers to derive alternate design structures for the target system data bases. [BLOCK 14]

Requirements Traceability Analyzer (3.2X)

TABLE FORMAT: This report compares one data base to another by tracing requirements (objects) from one data base to another. The trace is performed in both directions,

forwards and backwards. The traceability is performed using the TRACEKEY relationships in one data base to an object-name (usually a source document paragraph number, i.e., an identifier for the source of the requirement) in another data base. Requirements which do not trace either forward or backward are listed. The main report displays the requirement (object-name) of one data base on the left side with the requirement (object-name) in the second data base presented on the left side. The sources of the requirements are also displayed. This report is useful in tracing requirements from one data base to another. Applications include tracing higher level requirements represented in one data base model to the allocated requirements as represented in the second data base model. This report is a concise presentation of the traceability of requirements and provides completeness analysis of the traceability of requirements from one system model to another. [BLOCKS 13, 14, 15]

Resource Consumption Analysis (3.2)(5.1)

TABLE & MATRIX FORMAT: Displays resources consumed by a PROCESSOR; used by design engineers in evolving alternative designs in terms of resources used. [BLOCK 14]

Security Analysis Report (4.2)(5.1)

LIST, MATRIX, NARRATIVE & OUTLINE FORMAT: Displays security information about the objects; used to maintain consistency in defining the objects that are of a classified nature and are to be factored into the design of in the target system. [BLOCK 14]

Structure Report (3.2)(3.2X)(4.2)(5.1)

STRUCTURED-LIST FORMAT: Displays hierarchies of system objects where the levels of indenture represent the hierarchy of the objects. Hierarchies can be displayed to represent system structure, system flow, and dynamics. Hierarchies are based on the relationships defined between the objects such as subpart/part, consist/contained, receives/received, etc. The report is useful in evaluating the consistency and completeness of system hierarchies. Version 3.2X has user options which allow additional information about PROCESSES to be displayed immediately after each PROCESS in the report. Options include the display of data and control flow relationships, and memo, tracekey, and source information. In effect the 3.2X extensions combine some aspects of the Formatted Problem Statement Report into the Structure Report. This enhances the utility of the Structure Report, since more informational value is provided in a single report as an option to the user. [BLOCKS 4, 5, 6, 8, 9 10, 11, 12, 14]

Subset Analysis Report (4.2)(5.1)

MATRIX FORMAT: Displays information about SETS, such as hierarchical structure (using SUBSETS/SUBSET) and the interaction of SETS to other SETS along with INPUTS, OUTPUTS, and ENTITIES which make up the SET. Aids consistency and completeness checking of logical system data structures. [BLOCK 14]

Unit Structure (5.1)

STRUCTURED-LIST FORMAT: Displays hierarchy of system UNIT object as defined by the EQUIVALENT statement. Provides some automated completeness and consistency checking. [BLOCK 14]

Utilization Analysis Report (4.2)(5.1)

STRUCTURED-LIST & MATRIX: Displays information about PROCESSES such as the UTILIZES structure, and the interaction between PROCESSES (via subpart, utilizes) in a matrix format. Aids checking the consistency and completeness of the PROCESS structure. [BLOCKS 10, 14]

LOGICON

APPENDIX D

EXAMPLE ANALYZER REPORTS

The following 11 figures have been selected from a variety of Logicon requirements engineering projects to illustrate some of the reports described in this guidebook. These reports were genereated using version 3.2X of PSL/PSA (i.e., CADSAT: URL/URA). The figures included are as follows:

Figure 1	Input-PSL As-Is Source Listing	
Figure 2	Data Base Status report	(Logicon Extension)
Figure 3	Structure Report	(Logicon Modified)
Figure 4	Process Chain Report	
Figure 5	Data Process Report	
Figure 6	Contents Report	
Figure 7	Extended Picture Report	
Figure 8	Name-Gen & Formatted Problem Statement Reports	
Figure 9	Requirements Traceability Report	(Logicon Extension)
Figure 10	Formatted Problem Statement report	
Figure 11	Print Requirements Report	(Logicon Extension)

LOGICON

CDS Version 3.2R2 JUN 30, 19 16:42:38
LOGICON LEXINGTON VAX SYSTEM

a s - i s s o u r c e l i s t i n g

```
Lne  s t m t

310 >prc  process-cartographic-data;
311 >src s-1.0, s-3.1;
312 >desc;
313 >The Clustered Carto Processing System (CPS), will accept digitize
314 >linear data, perform various transformations on it, process it,
315 >validate it, perform automatic editing, and allow for cartographe
316 >interaction prior to generating the data in the proper format for
317 >inclusion in the Cartographic Data Base (CDB);
318 >sm  dn-interactive-capability;
319 >supp do-cartographics-applications,
320 >     sup-cartographics-applications;
321 >
322 >prc  do-cartographics-applications;
323 >supp accept-linear-digital-data,
324 >     val/correct-converted-cps-data,
325 >     perform-automatic-editing,
326 >     perform-interactive-editing,
327 >     perform-output-processing;
328 >
329 >prc  accept-linear-digital-data;
330 >syn  aclidida, input-proc;
331 >src  s-1.0, s-3.1, s-a1-1.0, s-a1-3.0;
332 >desc;
333 >The CPS is required to accept linear digital data from a variety
334 >of collection systems and in a variety of formats;
335 >sm  dn-data-input-on-mag-tape,
336 >    dn-dma-std-exchange-fmt,
337 >    dn-dma-lis-tape-format,
338 >    dn-ifass-tape-format,
339 >    dn-data-input-on-punched-cards;
340 >supp read-carto-tape-data,
341 >     convert-to-internal-format,
342 >     thin-cps-data,
343 >     transform-cps-data,
344 >     convert-cps-datum;
345 >
346 >prc  read-carto-tape-data;
347 >src  s-a1-3.0;
```

Figure 1. The As-Is Source Listing is one of eight reports produced when modifying the requirements data base. This example is an excerpt from the language statements entered for the CPS using the modifier command input-PSL.

LOGICON

JOINT SURVEILLANCE SYSTEM REV-PERIO 30 DATE 04/18/78 1425.6 est Tue

L ICON EXTENDED CADSAT version 3.2r1 ROCC SPECIFICATION D/ BASE STATUS REPORT

REQ. NO.	REQUIREMENT NAME	SYNONYM	TYPE	CAT	SEQN	STAT	NREV	SY	OS	SR	KW	TK	UD	RG	TT	PS	UU	MM	OK
3-7-1-2-3-a	radar-input-capacity-ti ing	rainca	memo	surv	4	COMPL	0	1	0	0	2	0	00	00	00	16	00	2	2
3-7-1-2-3-b	process-simulated-radar data	prsira	proc	surv	31	COMPL	0	1	0	0	1	0	00	00	00	13	00	0	2
	process-in-std-operatic s-only	prstop	proc	surv	32	COMPL	0	1	0	0	1	0	00	00	00	10	00	0	2
3-7-1-2-3-c	process-in-simulation-c er-lie	prsili	proc	surv	33	COMPL	0	1	0	0	3	0	00	00	00	10	00	0	2
3-7-1-2-3-d	process-in-simulated-ex rcise	prsiex	proc	surv	34	COMPL	0	1	0	0	1	0	00	00	00	10	00	0	2
3-7-1-2-3-3	segregate-simulated-dat	sesida	proc	surv	35	COMPL	0	1	0	0	1	0	00	00	00	10	00	0	2
3-7-1-2-3-f	radar-input-capacity-ti ing	rainca	memo	surv	4	COMPL	0	1	0	0	2	0	00	00	00	16	00	2	2
3-7-2-3-g	filter-radar-inputs	firame	proc	surv	5	COMPL	0	1	0	0	1	0	21	00	16	10	00	0	2
3-7-1-2-3-1	accept-sim-via-priority	acsipr	proc	surv	36	COMPL	0	1	0	0	2	0	00	00	00	10	00	0	2
3-7-1-2-3-1-1-a	accept-normal-radar-dat	acnora	proc	surv	6	COMPL	0	2	0	0	1	0	00	00	00	13	00	2	2
3-7-1-2-3-1-1-b	set-status-srch-iff	setsr	proc	surv	47	COMPL	0	1	0	0	1	0	11	00	00	10	00	0	2
	accept-sim-via-priority	acsipr	proc	surv	36	COMPL	0	1	0	0	2	0	00	00	00	10	00	0	2
3-7-1-2-3-1-1-c	map-srch-iff	masrif	proc	surv	7	COMPL	0	1	0	0	2	0	21	00	11	10	00	0	2
3-7-1-2-3-1-1-d	inputs-in-ptc-srch-iff	inptsr	memo	surv	6	COMPL	0	1	0	0	1	0	00	00	00	13	00	2	2
3-7-1-2-3-1-1-e	flag-emergency-codes	flemco	proc	surv	18	COMPL	0	1	0	0	1	0	11	00	12	10	00	0	2
3-7-1-2-3-1-1-f	acceptance-capacity-src iff	accsr	memo	surv	6	COMPL	0	1	0	0	1	0	00	00	00	13	00	2	2
3-7-1-2-3-1-1-g	limit-set-srch-iff-data	lissid	proc	surv	9	COMPL	0	1	0	0	1	0	21	00	11	10	00	0	2
3-7-1-2-3-1-1-h	limit-system-srch-iff-d a	lissif	proc	surv	10	COMPL	0	1	0	0	3	0	12	00	12	10	00	0	2
3-7-1-2-3-1-1-i	limit-system-srch-iff-d a	lissif	proc	surv	10	COMPL	0	1	0	0	3	0	12	00	12	10	00	0	2
3-7-1-2-3-1-1-j	count-srch-iff	cosrif	proc	surv	12	COMPL	0	1	0	0	1	0	31	00	14	10	00	0	2
	provide-srch-iff-alerts	prsrif	proc	surv	13	COMPL	0	1	0	0	1	0	00	00	00	12	00	0	2
	site-low-data-alerts	silода	proc	surv	14	COMPL	0	1	0	0	1	0	11	00	10	10	00	0	2
3-7-1-2-3-1-1-k	sys-nish-cond-chk-srch- iff	sncesi	proc	surv	16	COMPL	0	1	0	0	1	0	11	00	10	10	00	0	2
3-7-1-2-3-1-1-l	sys-overload-cond-chk-si ch-iff	socesi	proc	surv	17	COMPL	0	1	0	0	1	0	11	00	10	10	00	0	2
3-7-1-2-3-1-2	bias-adjustment-srch-iff	biadsr	proc	surv	51	COMPL	0	1	0	0	1	0	33	00	12	10	00	0	2

Figure 2. One purpose of the Data Base Status re rt developed by Logicon, is to compare the requirements data base contents against the source documentation. The left two columns of the repori display the requirements or red according to the sources of the requirements. This report is one of three ports which can also be used to display summary information on the contents of the requir ents data base. The right most columns provide a variety of statistics on the requirements displayed in the second column.

Figure 3. The Structure report displayed on the following three pages is one of three reports which show the functional hierarchy. This example report shows a preliminary functional hierarchy of the CPS first illustrated in Figure 1 using an indented format and includes the extended features developed by Logicon.

LOGICON

CDS Version 3.2R2 JUL 1, 19 0 14:41:55

LOGICON LEXINGTON VAX SYSTEM

process structure

count	(level or relationship)	names	reference
1	1	process-cartographic-data	s-1.0 s-3.1
		memo dn-interactive-capability	s-1.0 s-a1-4.11.3
		pfms clustered-carto-processing-sys	s-1.0 s-3.1
2	2	do-cartographics-applications	
		pfms applications-software	s-4.1.3 s-4.1.3.1 s-4.1.3.1.1 s-4.1.3.1.2
3	3	accept-linear-digital-data	s-1.0 s-3.1 s-a1-1.0 s-a1-3.0
		memo dn-data-input-on-mag-tape	s-a1-3.0
		memo dn-data-std-exchange-fmt	s-a1-3.1 s-a1a1-
		memo dn-data-lis-tape-format	s-a1-3.3 s-a1a4-
		memo dn-ifass-tape-format	s-a1-3.4 s-a1a5-
		memo dn-data-input-on-punched-cards	s-a1-3.0
		memo dn-data-std-production-fmt	s-a1-3.2 s-a1a2-
4	4	read-data-to-tape-data	s-a1-3.0
5	5	read-lis-geo/table-tapes	s-a1-3.3
		drvs lis-data	
		using lis-data-tape	s-a1-3.3
6	5	read-ifass-tapes	s-a1-3.4
		drvs ifass-header-data	
		using ifass-tape	s-a1-3.4
7	5	read-dma-exchange-tapes	s-a1-3.1
		drvs dma-std-exchange-data	
		using dma-std-exchange-tape	s-a1-3.1
8	5	read-dma-production-tapes	s-a1-3.2
		drvs dma-std-production-data	
		using dma-std-production-tape	s-a1-3.2
4	4	convert-to-internal-format	s-a1-3.2 s-a1-3.3 s-a1-4.1.1 s-a1a2-
		drvs cds-data	
		using dma-std-production-data	
		dma-std-exchange-data	

Figure 3 (cont.), Page 1 of 3

AD-A107 744 LOGICON INC LEXINGTON MA F/G 9/2
REQUIREMENTS ENGINEERING GUIDEBOOK. REQUIREMENTS ENGINEERING US--ETC(U)
JUL 80 D G SMITH DAA029-76-D-0100
UNCLASSIFIED FSD-R0022-VOL-II AIRMICS-80-8-2 NL

2 OF 2
AD A107 744

END
DATE FILMED
-82
DTIC

LOGICON

```
                 lis-data
                 ifass-header-data
10          4 thin-cps-data                    s-a1-4.3
              drvs thinned-cps-data
              using cps-data
11          4 transform-cps-data
              drvs transformed-cps-data
              using thinned-cps-data
12            5 trans-utm-projection
13            5 trans-polyconic-projection
14            5 trans-lambert-confrml-projectn
15            5 trans-polar-stereog-projection
16            5 trans-mercator-projection
17          4 convert-cps-datum                s-a1-4.1.4
                                               s-a1a6-
              drvs converted-cps-data
              using transformed-cps-data
18            5 convert-north-american-datum
19            5 convert-european-datum
20            5 convert-nrtn-amer/canada-datum
21            5 convert-tokyo-datum
22            5 convert-wgs-72-datum
23            5 convert-other-cps-datum
24        3 val/correct-converted-cps-data   s-a1-4.5
25          4 check-dataset-type/level
26          4 check-dataset-boundary
              memo product-dependent
27          4 thin-dataset                     s-a1-4.3
28          4 check-coords-crossing-boundary
29          4 check-feature-for-loops
30          4 check-feature-hierarchy
31          4 check-feature-header-for-sizes
32          4 check-areal-featur-for-closure
33          4 perform-linear-feature-checks
34        3 perform-automatic-editing
35          4 panel-datasets                   s-a1-4.8
36          4 merge-datasets                   s-a1-4.7
                                               s-a1-4.7.1
                                               s-a1-4.7.2
                                               s-a1-4.10
37          4 extract-data                     s-a1-4.9
38          4 plot-data                        s-a1-2.2
                                               s-a1-4.11.4
39        3 perform-interactive-editing      s-a1-4.11.3
                                             s-a4-4.0
40          4 perform-interac-edit-transfers   s-a1-4.11.3
                                               s-a4-4.0
41            5 perform-`to-edit`
42            5 perform-`from-edit`
43          4 perform-on-line-editing
44            5 register-digitizer-tablet
45            5 control-displays                 s-a1-4.11.1
46            5 edit-headers
```

Figure 3 (cont.), Page 2 of 3

```
47              5 edit-existing-linear-features
48                 6 find-feature
49                 6 display-header
50                 6 walk-feature
51                 6 digitize-stream-of-points
52                 6 digitize-single-point
53                 6 snap-to-point
54                 6 join-smoothly
55                 6 join-endpoint-to-string
56                 6 create-new-feature
57                 6 erase-feature-segments
58                 6 perform-othr-edit-lin-fea-fcts
59              5 digitize-new-linear-features     s-a1-4.11.2
60              5 escape-from-on-line-editing
61        3 perform-output-processing        s-a1-4.1.3
62           4 perform-datum-conversion
63           4 transform-projection
64           4 convert-format
65           4 write-output-tapes
66           4 validate-output-tapes
67     2 sup-cartographics-applications
         pfms system-software                    s-4.1.3
                                                 s-4.1.3.2
                                                 s-4.1.3.2.1
```

level	count	level	count	level	count	level	count	level	count
1	1	2	2	3	5	4	25	5	23
6	11								

Figure 3 (cont.), Page 3 of 3

CDS Version 3.2R2 JUN 25, 19 0 12:58:37

LOGICON LEXINGTON VAX SYSTEM

process chain

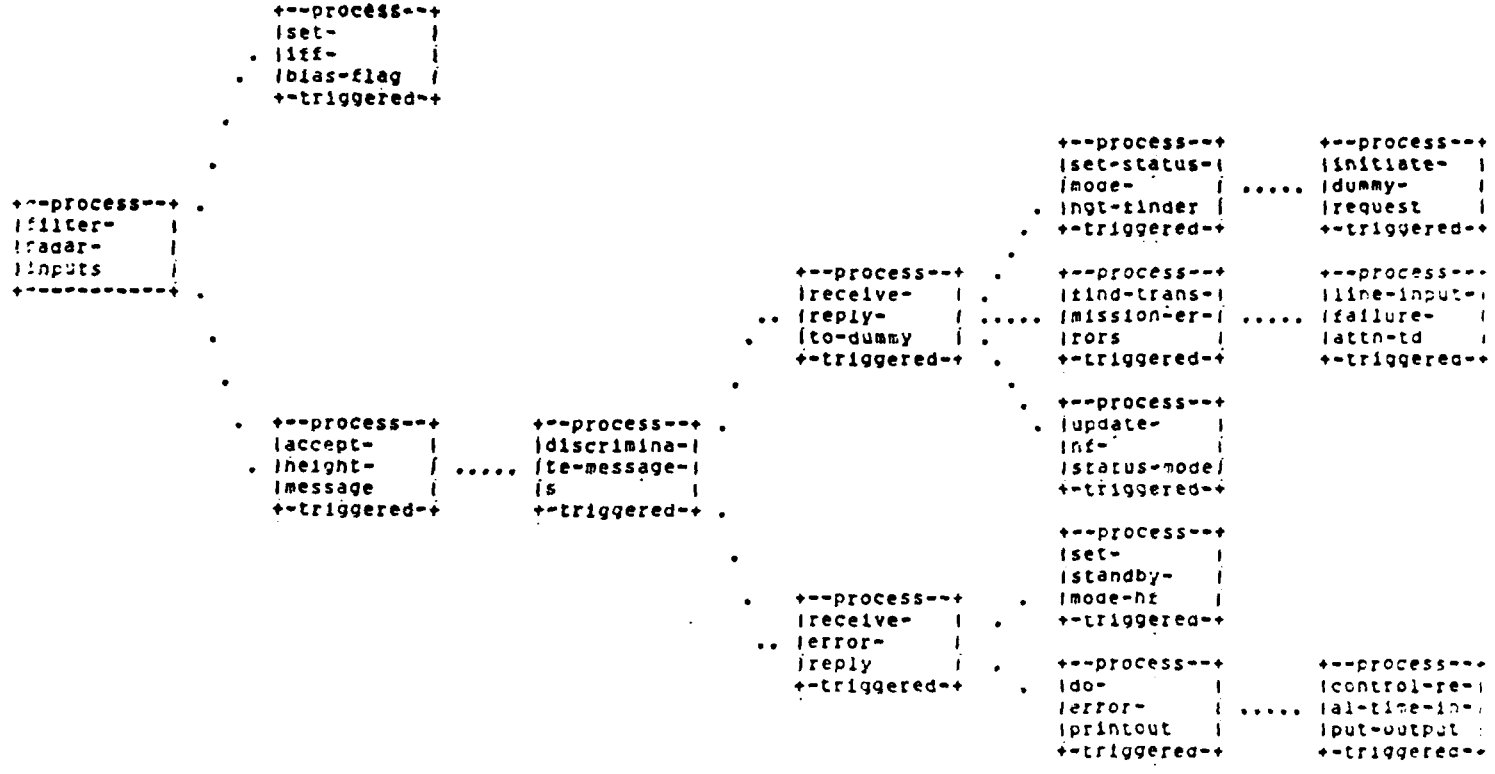


Figure 4. The Process Chain report can be used to display the sequence of functions, i.e. functional/control flow. This example excerpt from the Joint Surveillance System (JSS) application shows a sequence of functions (PROCESSES) beginning with filter-radar-inputs.

Figure 5. The Data Process Report displayed on the following two pages is one of several reports which show the interaction between selected functions and associated I/O. This example shows 20 I/O names (row names) and 27 functions/PROCESSES (column names) in a matrix. The interaction between the I/O and functions is displayed in the intersection of the matrix using the values shown immediately above the matrix.

LOGICON EXTENDED CADSAT version 3.2r1 04/01/78 1223.0

data process report

the rows are data names, the columns are process names.

	row names	
1	selected-acquisition-report	output
2	comprehensive-data-base	set
3	life-cycle-cost-report	output
4	allocation-cost-dollar-value	element
5	repair-description	group
6	component-item-record	group
7	end-item-depot-record	group
8	share-of-cost	element
9	percent-of-total-time	element
10	opt-repair-lev-anal-report	output
11	number-of-units	element
12	minimum-cost-value	element
13	depot-cost-dollar-value	element
14	intermediate-cost-dollar-value	element
15	discard-cost-dollar-value	element
16	minimum-cost-alternative	element
17	facility-cost	group
18	inventory-cost	group
19	initial-training-cost	group
20	life-cycle-period	group

	column names	
1	management-information-system	process
2	user-functions	process
3	reporting-capabilities	process
4	life-cycle-cost-analysis	process
5	development-cost-estimating	process
6	operations-cost-estimating	process
7	optimum-repair-level-analysis	process
8	maintenance-cost-estimating	process
9	financial-planning-tracking	process
10	generate-cpr	process
11	generate-cfsr	process
12	generate-estimated-costs	process
13	generate-cssr	process
14	schedule-planning-and-tracking	process
15	generate-cdrl-schedule	process
16	generate-network-data	process
17	generate-milestone-schedule	process
18	generate-manpower-schedule	process
19	generate-ecp-status	process
20	requirements-analysis	process
21	requirements-relation-analysis	process
22	requirements-evaluation	process
23	system-performance-analysis	process
24	system-completeness-assessment	process
25	system-consistency-analysis	process
26	traceability-analysis	process
27	configuration-accounting	process

Figure 5 (cont.), Page 1 of 2

LOGICON

data process interaction matrix

(i,j) value	meaning
r	row i is received or used by column j (input)
u	row i is updated by column j
d	row i is derived or generated by column j (output)
a	row i is input to, updated by, and output of column j (all)
f	row i is input to and output of column j (flow)
1	row i is input to and updated by column j
2	row i is updated by and output of column j

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	i		d		i					i		d				i					i					i					i
2	i		r		i		r			ri	r	r	r			i	r				i							r			i
3	i			d	i		d			i						i					i					i					i
4	i				i		d			i						i					i					i					i
5	i				i		r			i						i					i					i					i
6	i				i		r			i						i					i					i					i
7	i				i		r			i						i					i					i					i
8	i				i		d			i						i					i					i					i
9	i				i		d			i						i					i					i					i
10	i				i		d			i						i					i					i					i
11	i				i		d			i						i					i					i					i
12	i				i		d			i						i					i					i					i
13	i				i		f			i						i					i					i					i
14	i				i		f			i						i					i					i					i
15	i				i		f			i						i					i					i					i
16	i				i		f			i						i					i					i					i
17	i				i		r			i						i					i					i					i
18	i				i		r			i						i					i					i					i
19	i				i		r			i						i					i					i					i
20	i				i		r			i						i					i					i					i

Figure 5 (cont.), Page 2 of 2

LOGICON

LOGICON EXTENDED CADSAT version 3.2r1 04/01/78 038.1

contents report

```
1*    1 input-set (set)
 1      2   system-acquisition-plan-init (input)
 2        3     contract-funding-for-fy  (element)
 3        3     contract-appropriation (element)
 4        3     appropriation-identification (group)
 5        3     funding-authorization-to-date (group)
 6        3     contract-funding-profile (group)
 7         4       unliquidated-commitments (group)
 8          5        actual-to-date (element)
 9          5        projected-by-period (group)
10          5        at-completion (element)
11         4       accrued-expenditures (group)
12          5        actual-to-date (element)
13          5        projected-by-period (group)
14          5        at-completion (element)
15         4       total-costs (group)
16          5        actual-to-date (element)
17          5        projected-by-period (group)
18          5        at-completion (element)
19         4       forecast-billings (group)
20          5        projected-by-period (group)
21          5        at-completion (element)
22        3     budgeted-cost-work-scheduled (group)
23        3     undistributed-budget (group)
24        3     production-vs-rate (element)
25        3     program-name-number (group)
26         4       contract-program-name  (element)
27         4       contract-number (element)
28        3     estimated-unpriced-work (element)
29        3     contract-budget-baseline (element)
30        3     bcws-six-months-forecast (group)
31        3     bcws-period-forcast (group)
32        3     prime-item-quantity (element)
```

Figure 6. The Contents report is one of three reports which can be used to display selected levels (branches) of the I/O hierarchical structure. The report format is in the form of an indented listing where the hierarchy levels are denoted by the number preceding the I/O name.

CDS Version 3.2R2 JUN 30, 1 1 17:51:06

LOGICON LEXINGTON VAX SYSTEM

extended picture

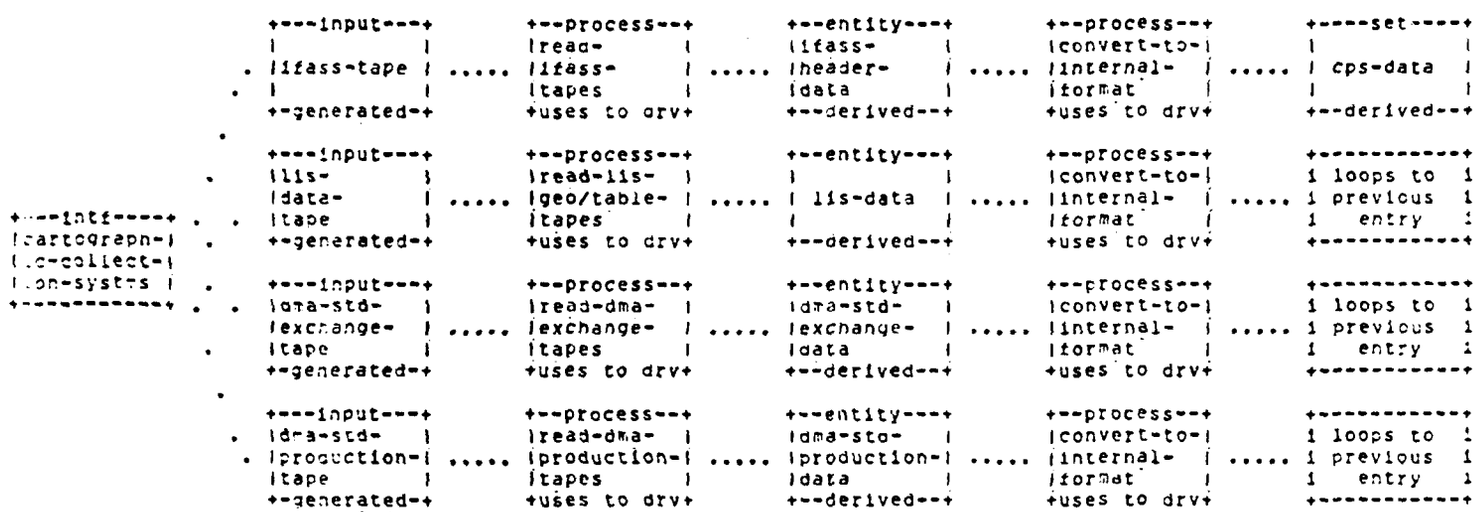


Figure 7. Information flow into, within, and out of the target system can be shown using the Extended Picture report. This example from the CPS requirements data base (Figure 1) shows the flow of four INPUTS generated from a single INTERFACE, cartographic-collection-systms. The report continues to additional pages until all defined flow is presented.

LOGICON

CDS Version 3.2R2 JUL 2, 1990 10:31:51
LOGICON LEXINGTON VAX SYSTEM

name generation

parameters for: ng

```
print punch empty selection='process and key=tracking' order=bytype

 1    active-tracking                    process
 2    aid-system-in-trking-function      process
 3    air-target-detection               process
 4    allocate-surv-task-to-surv-ops     process
```

CDS Version 3.2R2 JUL 2, 1990 10:32:06
LOGICON LEXINGTON VAX SYSTEM

formatted problem statement

```
 1 process                              air-target-detection;
 2     synonyms are:  airtardet;
 3     see-memo:      py-air-target-surv-area,
 4                    py-air-traffic-environment,
 5                    pf-as-atmospheric-anomalies,
 6                    dn-as-oper-selectble-surv-mode;
 7     keywords:      tracking,
 8                    radar,
 9                    detection;
10     trace key:     m-3.1.1.1.1.1.2.c.1,
11                    m-3.3.3.1.2.1,
12                    m-3.3.3.1.2-table-48,
13                    m-3.3.3.1.1;
14     subparts are:  obtain-height-data,
15                    smooth-height-observations,
16                    quantize-height-before-storing;
17     part of:       target-detection;
18     source is:     g-3.1.1.1.1.2.1.4,
19                    g-3.1.1.1.1.2.1.4.c.1;
```

Figure 8. By combining the features of two reports the analyst can locate and display selected information on a requirement which may be changed. Using the Name-Gen report shown at the top, a list of functions (PROCESSES) with a keyword of tracking can be extracted. The analyst can next produce a Formatted Problem Statement report for each of the 4 function names or a single function name such as the report shown at the bottom.

LOGICON

E3A PROGRAM LOGICON REQUIREMENTS TRACEABILITY REPORT 11-FEB-80

DATA BASES TRACED:
e3amsiss to radar

DB1-REF	REQUIREMENT NAME	DB2-REF	REQUIREMENT NAME
m-3.1	dedicated-air-operation	r-1.2.a	air-surveillance-type-II
m-3.1	dedicated-air-operation	r-3.1.1.1.2.1	
m-3.1	dedicated-maritime-operation	r-3.1.1.1.2.2.5.1.b	py-ms-surveillance-volume
m-3.1	dedicated-maritime-operation	r-3.1.1.1.2.2.5.1.b	ms-detect-maritime-targets
m-3.1	dedicated-maritime-operation	r-1.2b	maritime-surveillance-type-viii
m-3.1	dedicated-maritime-operation	r-3.1.1.1.2.2.5.1	maritime surveillance-type-viii
m-3.1.1.1.1	py-clear-and-ecm-environments	r-3.1.1.1.1.e	py-as-non-ecm-environment
m-3.1.1.1.1	py-clear-and-ecm-environments	r-3.1.1.1.1.f	py-as-ecm-environment
m-3.1.1.1.1	py-clear-and-ecm-environments	r-3.1.1.1.1.e	py-as-non-ecm-environment
m-3.1.1.1.1	py-clear-and-ecm-environments	r-3.1.1.1.1.f	py-as-ecm-environment
m-3.1.1.1.1.a	dedicated-air-operation	r-1.2.a	air-surveillance-type-II
m-3.1.1.1.1.a	dedicated-air-operation	r-3.1.1.1.2.1	
m-3.1.1.1.1.c	dedicated-maritime-operation	r-3.1.1.1.1.2.2.5.1.b	py-ms-surveillance-volume
m-3.1.1.1.1.c	dedicated-maritime-operation	r-3.1.1.1.1.2.2.5.1.b	ms-detect-maritime-targets
m-3.1.1.1.1.c	dedicated-maritime-operation	r-1.2.b	maritime-surveillance-type-viii
m-3.1.1.1.1.c	dedicated-maritime-operation	r-3.1.1.1.2.2.5.1	maritime-surveillance-type-viii
m-3.1.1.1.1.1.2	process-radar-returns	r-3.1.1.2.1.1	as-resolve-range-ambiguities
m-3.1.1.1.1.1.2	target-detection	r-3.1.1.1	detect-and-report-air-targets
m-3.1.1.1.1.1.2	target-detection	r-3.1.1.1	as-detect-range
m-3.1.1.1.1.1.2	target-detection	r-3.1.1.1	as-report-range
m-3.1.1.1.1.1.2	target-detection	r-3.1.1.1	as-detect-azimuth
m-3.1.1.1.1.1.2	target-detection	r-3.1.1.1	as-report-azimuth
m-3.1.1.1.1.1.2	target-detection	r-3.1.1.1	as-detect-elevation-angle
m-3.1.1.1.1.1.2	target-detection	r-3.1.1.1	as-report-elevation-angle
m-3.1.1.1.1.1.2	target-detection	r-3.1.1.1	as-detect-ecm
m-3.1.1.1.1.1.2	target-detection	r-3.1.1.1	as-minimize-effect-on-surv-vol
m-3.1.1.1.1.1.2.a.a	py-air-surveillance-volume	r-3.1.1.1.1.1.c	py-as-surveillance-volume
m-3.1.1.1.1.1.2.a.a	position-antenna-beam	r-3.1.1.1	detect-and-report-air-targets
m-3.1.1.1.1.1.2.a.1	position-antenna-beam	r-3.1.1.1	as-detect-range
m-3.1.1.1.1.1.2.a.1	position-antenna-beam	r-3.1.1.1	as-report-range
m-3.1.1.1.1.1.2.a.1	position-antenna-beam	r-3.1.1.1	as-detect-azimuth
m-3.1.1.1.1.1.2.a.1	position-antenna-beam	r-3.1.1.1	as-report-azimuth
m-3.1.1.1.1.1.2.a.1	position-antenna-beam	r-3.1.1.1	as-detect-elevation-angle
m-3.1.1.1.1.1.2.a.1	position-antenna-beam	r-3.1.1.1	as-report-elevation-angle
m-3.1.1.1.1.1.2.a.1	position-antenna-beam	r-3.1.1.1	as-detect-ecm

Figure 9. The Requirements Traceability Analyzer report, developed by Logicon, is used to trace requirements between different requirements data bases such as DB1 (e3amsiss) and DB2 (radar) shown above. For example, the requirement target-detection which is a requirement identified in the source m-3.1.1.1.1.1.2 of DB1 (source document: e3amsiss) traces to nine requirements of the second source document (radar specification, paragraph r-3.1.1.1) as represented in the second requirement's data base (DB2).

LOGICON

CDS Version 3.2R2 JUL 2, 1980 14:26:02

LOGICON LEXINGTON VAX SYSTEM

formatted problem statement

```
 1 event                                                maximum-loads-  st;
 2     description;
 3
 4 1.0  Test Objectives
 5
 6      The objective of the Maximum Loads Test is to verify the capab ity
 7 of the application software to operate with the maximum number of s tem
 8 tracks, active flight plans, simultaneous intercepts, radar input and
 9 track telling.  System response times will be verified while unde the
10 loads specified.
11
12 2.0  Test Approach
13
14      A simulation tape will be generated which will provide an incr en-
15 tal buildup of the processi g loads to verify each of the system ca, ci-
16 ties specified in Table VIII of the system specification (r-).   S) tem
17 operation will be demonstrated with the following maximum loads:
18
19      (a)  Maximum number of active flight plans
20      (b)  Maximum number of search tracks
21      (c)  Maximum number of passive tracks
22      (d)  Maximum number of simultaneous intercepts
23      (e)  Maximum number of radar inputs
24      (f)  Maximum number of track tell messages.
25
26 3.0  Acceptance Criteria
27
28      The system shall continue to function through each of  the  max mum
29 load  increments  as  indicated  by  the situation and tabular displ ys.
30 System response times will be as specified under the design loads.;
31     keywords:      tracks,
32                    flight-plans,
33                    intercepts,
34                    radar-inputs,
35                    track-tell-messages,
36     attributes are:
37          initial-date   jun-06-1979,
38          revision-date  jan-17-1980,
39          status         incomplete;
40     on inception of:
41                    perform-application-tasks;
42     source is:     r-table-viii;
```

Figure 10. The Formatted Problem Statement report can be used to display test plans and procedures as illustrated for a surveillance syste test above. The EVENT object (line 1 can be used to define a single test point on a sys em flow or a collection of points which define a test case. The test can be associat d with a function (PROCESS) as shown in lines 44-45.

PTSS PROJECT LOGICON SPECIFICATION LISTING 6-JUN-80

DATA BASE: aptss

1.0 PRODUCTION-TERMINAL-SUPT-SYSTM
The Production Terminal Support System (PTSS) shall have the functional capabilities and performance factors given in the following specification.
REF: N/A

1.1 SYSTEM-CONTROL
The PTSS shall provide a control capability to govern the use of system resources.
REF: N/A

1.1.1 LOGIN-CONTROL
The PTSS shall provide control of the user login, logout, and user access control.
REF: N/A

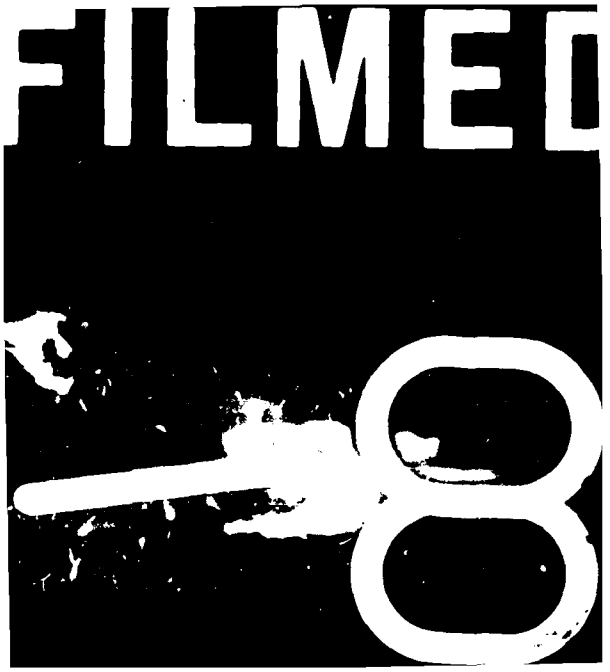
1.1.1-1 DN-LOGIN-ONCE-PER-TERMINAL
A user shall not have to login to each partition - initial login shall be sufficient.
REF: s-196,g-236

1.1.1-2 DN-LIMIT-ONE-USER-PER-TERM
Only one user can be logged in on a physical terminal at any one time.
REF: s-196,g-236

1.1.1-3 PF-LOGIN-RESPONSE-REQ
The PTSS shall have a maximum average login command response time of 2.0 seconds and a maximum worst case response time of 5.0 seconds which shall not occur more than 5% of the time during prime shift hours.
REF: s-327,g-364

1.1.1.1 LOGIN-LOGOUT-CONTROL-FUNCTIONS
The PTSS shall provide control of the user login/logout process.
REF: N/A

Figure 11. The Print Requirements report displays the requirements data base information in a manner which is more typical of specification documention. The above report, excerpted from a longer report, shows the functional hierarchy using the node numbers of the hierarchy for paragraph numbers and includes text, source document references, and constraints associated with each function as maintained in the requirements data base.


FILME
8